École Polytechnique de l'Université de Nantes
Département INFORMATIQUE
Quatrième année

PROJET TRANSVERSAL

ANNÉE UNIVERSITAIRE 2007/2008

« Mise à disposition d'une interface ODBC pour n'importe quel type de source de données »

Rapport Final

Présenté par Fabien COCHETEUX & Nicolas DAVID
le : 26 mai 2008

Jury :
Coordinateur : Jean-Pierre GUÉDON
Tuteur entreprise : Hervé GUÉRIN
Tuteur enseignant : Nicolas NORMAND

Table des matières

I Bibliographie & Analyse des besoins 8

1 Document de spécification technique des besoins 10

1.1 Fondements du projet 10
1.1.1 But du projet 10
1.1.2 Personnes et organismes impliqués 11
1.1.3 Utilisateurs du produit 11
1.2 Contraintes sur le projet 11
1.2.1 Contraintes non négociables 11
1.2.2 Glossaire et conventions de dénomination 13
1.2.3 Faits et hypothèses utiles 14
1.3 Besoins fonctionnels 14
1.3.1 Étendue du travail 14
1.3.2 Analyse orientée objet 15
1.3.3 Exigences fonctionnelles et techniques 18
1.4 Besoins non fonctionnels 19
1.4.1 Ergonomie et convivialité du produit 19
1.4.2 Facilité d'utilisation et facteurs humains 19
1.4.3 Fonctionnement du produit 20
1.4.4 Maintenance du produit 21
1.4.5 Calcul des coûts 21
1.4.6 Planning prévisionnel 21
1.5 Autres aspects du projet 22
1.5.1 Tâches à faire pour livrer le système 22
1.5.2 Contrôle final de qualité sur le site 22
1.5.3 Manuel utilisateur et formations 23

2 Etude bibliographique du domaine 24

2.1 Documentation sur les drivers 24
2.1.1 Qu'est ce qu'un driver ? 24
2.1.2 Les différents types de drivers établissant une connexion avec un SGBD 26

2.1.3 Les drivers JDBC 27
2.2 Licences libres 32
2.2.1 Domaine public 33
2.2.2 Licences protectrices (Copyleft) 33
2.2.3 Licences non protectrices 34
2.3 Sites de dépôt de projet 37
2.3.1 SourceForge 37
2.3.2 FreshMeat 39
2.3.3 Autres sites 40
2.4 Gestion des versions 41
2.4.1 CVS 41
2.4.2 SVN 44

II Modélisation 48

3 Complément de bibliographie 50
3.1 Rappels 50
3.1.1 Driver 50
3.1.2 ODBC 51
3.1.3 JDBC 52
3.2 Compléments d'information 52
3.2.1 Bridge JDBC-ODBC 52
3.2.2 Gateway ODBC-JDBC 54
3.2.3 JNI 56

4 Recherche de solution 59
4.1 ODBC-JDBC 59
4.1.1 Objectif 59
4.1.2 Résultat 60
4.2 ODBC 60
4.2.1 Objectif 60
4.2.2 Résultat 61

5 Dossier de conception 62
5.1 Vue générale de l'architecture du projet 62
5.1.1 Contexte du projet 62
5.1.2 Modules à implémenter 64
5.2 Conception des différentes parties 67
5.2.1 Le pilote ODBC 67
5.2.2 Réalisation d'un logiciel de substitution 71

5.2.3 Le *thread* serveur 71
5.2.4 Exécutable d'installation 74
5.2.5 Tests d'intégrité et validation du projet 74
5.3 Organisation du temps de travail 75
5.3.1 1er élément 75
5.3.2 2e élément 76
5.3.3 3e élément 76
5.3.4 4e élément 76
5.3.5 5e élément 77
5.3.6 6e élément 77

III Réalisation 80

6 Notre projet et ses objectifs 82
6.1 Rappel du travail précédemment effectué 82
6.1.1 Étape « Bibliographie & Analyse des besoins » 82
6.1.2 Étape « Modélisation » 83
6.2 Travail prévu pour la suite du projet dans le cadre de l'étape de réalisation 83
6.3 Point critique du processus d'implémentation 83

7 Résultats obtenus 84
7.1 Utilisation du programme compilé fourni 84
7.1.1 Installation automatique 84
7.1.2 Installation manuelle 86
7.2 Utilisation du code fourni et d'outils supplémentaires 87
7.2.1 Affichage des avertissements 88
7.2.2 Problème lors de la création du DNS 88
7.2.3 Problème d'appel de fonctions 91

8 Solutions envisagées 94
8.1 Côté contraignant d'ODBC 94
8.1.1 Un manque d'informations 94
8.1.2 Une technologie difficile à implémenter 95
8.2 Solutions envisageables pour l'avenir 95
8.2.1 Poursuivre avec ODBC 95
8.2.2 Utiliser un pilote ODBC déjà existant 96
8.2.3 Utilisation d'une autre API 97
8.2.4 Utiliser un fichier de données intermédiaire 98

8.3 Développement d'une solution : l'exportation de données au format CSV . . . 98
8.3.1 Cahier des charges . . . 99
8.3.2 Complément bibliographique . . . 100
8.3.3 Conception . . . 102

IV Dimension commerciale 109

9 Présentation de l'analyse 110
9.1 Présentation de l'entreprise . . . 110
9.2 Présentation du projet . . . 110
9.3 Problématique . . . 110

10 Analyse commerciale du projet 112
10.1 Objectifs de l'étude . . . 112
10.2 Présentation . . . 112
10.3 Réflexion marketing sur l'exploitation commerciale du projet . 113
10.3.1 Étude de marché . . . 113
10.3.2 Stratégie commerciale . . . 116

Annexes 121

Etude bibliographique 121

Utilisation d'un driver ODBC sous Access 97 126

Rapports hebdomadaires 131
Étape « Bibliographie & Analyse des besoins » . . . 132
Étape « Modélisation » . . . 143
Étape « Réalisation » . . . 150

Bibliographie 163

Table des figures

1.1 Schéma de l'ensemble fonctionnel 13
1.2 Diagramme de cas d'utilisation 16
1.3 Esquisse du diagramme de classes 16
1.4 Diagramme de séquences 17

2.1 Schéma d'un driver JDBC de type 1 [9] 28
2.2 Schéma d'un driver JDBC de type 2 [9] 29
2.3 Schéma d'un driver JDBC de type 3 [9] 30
2.4 Schéma d'un driver JDBC de type 4 [9] 31
2.5 Page d'accueil de SourceForge.net 38
2.6 Page d'accueil de FreshMeat 39

3.1 Schéma d'un pont JDBC-ODBC [6] 53
3.2 Schéma d'un driver ODBC-JDBC de type 1 [21] 55
3.3 Schéma de communications avec JNI [22] 56

5.1 Schéma général, représentant l'architecture du projet 63
5.2 Diagramme UML de séquences 65
5.3 Diagramme UML de classes, pour le pilote ODBC 68
5.4 Diagramme UML de classes, pour le *thread* serveur 72
5.5 Planning prévisionnel de la phase de réalisation 78

7.1 Driver reconnu 85
7.2 Base de registre 86
7.3 ODBC initialisation 87
7.4 Compilation 88

8.1 Schéma général 99
8.2 Diagramme de cas d'utilisation 101
8.3 Schéma de l'ensemble fonctionnel 103
8.4 Diagramme de classes 105

10.1 Etape 1 du chargement d'un driver ODBC sous *Access 97* . . 127

10.2 Etape 2 du chargement d'un driver ODBC sous *Access 97* . . 127
10.3 Etape 3 du chargement d'un driver ODBC sous *Access 97* . . 128
10.4 Etape 4 du chargement d'un driver ODBC sous *Access 97* . . 129
10.5 Etape 5 du chargement d'un driver ODBC sous *Access 97* . . 130

Dans le cadre du parcours ingénieur à l'École Polytechnique de l'Université de Nantes, doit se dérouler un projet transversal en collaboration avec une entreprise.

Le projet qui nous est proposé est celui de M. Hervé Guérin.
Le pilotage est assuré par M. Nicolas Normand.
La coordination est assurée par M. Jean-Pierre Guédon.

Le rapport du projet se découpe, en trois phases différentes :
- Bibliographie et analyse des besoins
- Modélisation
- Réalisation

A la fin du rapport, seront présentes nos références bibliographiques et nos annexes, dans lesquelles figurent nos fiches de suivi.

Première partie

Bibliographie & Analyse des besoins

Le but de cette partie est de délimiter les contours du projets, et de rassembler l'ensemble des connaissances nécessaires pour la suite du projet.

Cette bibliographie doit nous permettre d'aboutir sur le choix du type de driver que nous utiliserons dans la suite du projet.

Les livrables contenus pour cette étape sont une étude bibliographique « métier » du domaine et la rédaction d'un cahier des charges (documentation de spécification technique des besoins).

Chapitre 1

Document de spécification technique des besoins

Ce cahier des charges est inspiré du canevas décrit dans le document "*Volere*" (provenant du serveur pédagogique du département informatique).

Il contiendra notamment une analyse orientée objet (diagrammes UML pour l'analyse) et un calendrier de travail. Cette partie est un cahier des charges précis, directement lié au besoin exprimée par l'utilisateur. Il sera le point de départ de la deuxième étape.

1.1 Fondements du projet

Avant de commencer à parler des aspects techniques du projet, il est important de bien le présenter, c'est-à-dire de définir le but et les personnes impliquées dans les enjeux de ce dernier.

1.1.1 But du projet

Présenter le but d'un projet, c'est positionner ce projet dans le contexte auquel il est lié, et définir l'objectif auquel il devra répondre.

Contexte du projet

Dans le cadre du projet de lancement d'une *start-up*, M. Guérin compte développer un logiciel permettant la gestion de données dans une SGBD propriétaire.

Objectif du projet

L'objectif de ce projet est d'ajouter une fonctionnalité au logiciel développé par M. GUÉRIN.

Plus précisément, nous devons réaliser un driver capable d'exporter des données du logiciel de M. GUÉRIN vers un quel outil de requêtes du marché (*Business Objects*, *Cognos*, *Access*, *Excel*, ...).

Le logiciel de M. GUÉRIN étant encore en développement, nous nous contenterons, de consulter des données contenues dans des fichiers plats quelconques, depuis un logiciel permettant de le substituer.

1.1.2 Personnes et organismes impliqués

Ce projet a été proposé par M. GUÉRIN à *Polytech'Nantes* pour être traité dans le cadre des projets transversaux de 4ème année du département Informatique. Nous sommes donc contributeurs du projet.

M. NORMAND, en tant que tuteur enseignant du projet transversal, nous soutient et contrôle notre travail.

M. GUÉDON, également enseignant à *Polytech'Nantes*, sera pris comme tiers pour coordonner l'évaluation du travail effectué.

1.1.3 Utilisateurs du produit

La première finalité de cette brique logiciel est d'être intégrée au projet de M. GUÉRIN. Ce travail sera proposé en libre téléchargement sur un site de centralisation de projet libre de droit, tel que SourceForge.net.

Notre projet sera donc développé dans l'optique d'être rendu réutilisable.

1.2 Contraintes sur le projet

Les contraintes du projet permettent de cerner comment le produit final doit s'intégrer dans le monde réel.

Le produit doit notamment avoir à s'interfacer avec des logiciels déjà présents. Il doit aussi devoir être prêt à une date donnée.

1.2.1 Contraintes non négociables

Cette section décrit les contraintes sur la manière de satisfaire les exigences qui influencent la conception finale du produit.

Contraintes sur la conception de la solution

Toutes les sources devront être sauvegardées sur un dépôt de type *CVS/SVN*.

Le produit rendu sera sous la forme d'un DLL et d'une documentation décrivant comment l'utiliser. Nous remettrons les sources commentées du driver, développées en *JAVA* sous *Eclipse*.

Le driver devra être utilisable dans un projet commercial. Toutes les sources déjà existantes que nous utiliserons seront donc sous licence non virale. Afin de déterminer la licence sous laquelle le driver sera distribué, nous y joindrons la synthèse de notre recherche sur les licences.

Environnement de fonctionnement du système actuel

Les contraintes qui nous ont été spécifiées permettent de définir l'environnement de fonctionnement du système à construire. Ainsi, l'ensemble logiciel/driver devra fonctionner sur les systèmes d'exploitation *Windows XP* et *Windows Vista*.

Applications partenaires (Fig. 8.1)

Les contraintes spécifiées permettent de définir les applications partenaires du logiciel, c'est-à-dire celles avec lesquelles le produit doit collaborer.

Tout d'abord, l'utilisateur communique avec l'interface ODBC, présent dans les versions récentes de *Access* et de *Excel*. Le driver devra donc être (au moins) compatible avec ces deux logiciels, et ceux définis dans la spécification.

Ensuite, « de l'autre côté » de l'application (côté structure de données), on utilise une interface qui sera créée par M. Guérin. Comme spécifié sur le schéma, le driver devra donc être compatible avec la structure de données existante.

Contraintes sur la publication des informations

Cette section va nous permettre de donner les contraintes spécifiées quant à la publication des informations concernant le projet transversal.

Tout d'abord, tous les documents écrits devront être rédigés et compilés avec le système de traitement de texte LaTeX. On en fera de même pour les soutenance orales : leur support sera réalisé avec LaTeX accompagné du package *Beamer*, qui permet de réaliser des présentations en gardant le style de LaTeX.

Ensuite, un compte-rendu (hebdomadaire) devra être rendu en fin de chaque semaine. Il détaillera les activités réalisées durant la semaine passée,

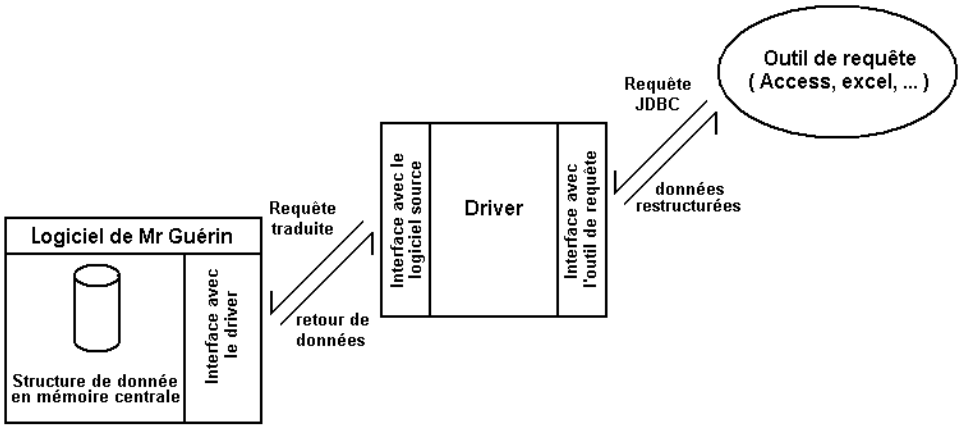


FIG. 1.1 – Schéma de l'ensemble fonctionnel

et donnera les activités prévues pour la semaine suivante. Devront aussi être rendus :

- les trois rapports d'étapes (bibliographie & analyse des besoins, réalisation, conception)
- un rapport final reprenant les précédentes étapes
- ainsi qu'un début de bibliographie et d'une ébauche de cahier des charges, à la fin du mois d'octobre.

Enfin, on s'accordera sur une diffusion totale des discussions entre chaque membre impliqué dans le projet transversal : les étudiants, le tuteur entreprise et le tuteur enseignant.

1.2.2 Glossaire et conventions de dénomination

Cette section va nous permettre de donner le sens des abréviations et des dénominations, qui seront utilisées tout au long de ce rapport.

JDBC Java database connectivity

ODBC Open database connectivity

ADO ActiveX Data Object

RMI Remote Method Invocation

SGBD Système de Gestion de Bases de Données (dans notre cas, la structure de données couplée au logiciel d'exploitation)

1.2.3 Faits et hypothèses utiles

Cette section va nous pemettre de définir des faits influançant le produit à développer, ainsi que des hypothèses utiles pour la suite du projet.

Facteurs influançant le produit

Un premier facteur influançant le produit est la non-compatibilité de *ODBC* avec *JAVA*. Pour contourner cet obstacle, on peut utiliser *JDBC*, construit sur le langage *JAVA*.

Hypothèses faites par l'équipe

On considérera que la structure de données, sur laquelle notre driver travaillera, sera existante et que nous pourrons l'utiliser afin de stocker des données et de traiter des requêtes sur celles-ci.

1.3 Besoins fonctionnels

Les exigences fonctionnelles constituent le sujet principal et la matière fondamentale pour la création du système. Elles sont mesurées par des moyens concrets comme les valeurs de données, la logique des processus décisionnels ou les algorithmes.

Ces exigences représentent les activités que le produit doit assurer.

1.3.1 Étendue du travail

La situation actuelle

Le logiciel développé par M. Guérin, sur lequel viendra se greffer notre travail est encore en cours de développement.

Nous disposons cependant des informations suivantes pour pouvoir réaliser notre driver :

- Le logiciel permettra à l'utilisateur de saisir et de visualiser des données organisées hiérarchiquement
- Ces données seront stockées en mémoire centrale sous forme de base de données non relationnelle
- Le logiciel sera interfaçable avec le driver que nous devons créer. Il se chargera de lire les données en mémoire centrale pour les envoyer au driver. Nous n'avons donc pas à nous soucier de la structure de la base de données.

- Le logiciel est développé dans l'optique d'être commercialisé. Le composant que nous devons produire doit donc pouvoir être intégré dans un projet commercial.
- Le logiciel tournera exclusivement sous *Windows* (*XP*/*Vista*).

Les données récoltées via notre driver seront retransmises à des systèmes de gestion de bases de données (SGBD) classiques ainsi que des outils de traitement de données (*Access*, *Excel*, *Business Object*, *Cognos*, *Oracle*, ...). Ces outils pourront permettre aux utilisateurs du logiciel développé par M. Guérin, d'appliquer leurs propres algorithmes pour traiter et analyser les données ainsi exportées.

Contexte du travail

Comme vu précédemment, le driver que nous devons créer doit servir d'intermédiaire entre le logiciel de M. Guérin et des SGBD classiques.

Pour réaliser ce driver, nous partirons d'un projet *Open Source* déjà existant capable de se connecter aux SGBD classiques. Nous n'aurons donc pas à nous occuper de cette fonctionnalité du driver. En revanche, nous étudierons comment réaliser la jonction entre ce driver et le logiciel dont on veut extraire les données. Nous pouvons déjà identifier deux étapes : la transmission des requêtes formulées par le SGBD et la récupération des résultats.

Nous chercherons à réaliser une interface permettant aux deux logiciels de communiquer.

Divisions du travail en évènements métier

S'appuyant sur les exigences citées ci-dessus, voici une liste des tâches dans lesquelles notre driver entre en jeu :

1. A la demande du SGBD, le driver établit les connexions avec le SGBD et le logiciel dont on veut extraire les données
2. En les ayant préalablement traduites, le driver retransmet les requêtes envoyées par le SGBD au logiciel dont on veut extraire les données
3. En réponse à une requête, le logiciel transmet des données au driver qui, après en avoir adapté le contenu, les renvoie au SGBD

1.3.2 Analyse orientée objet (Fig. 1.2, Fig. 1.3 & Fig. 1.4)

Cette sous-section présente différents diagrammes UML permettant une analyse orientée objet du projet.

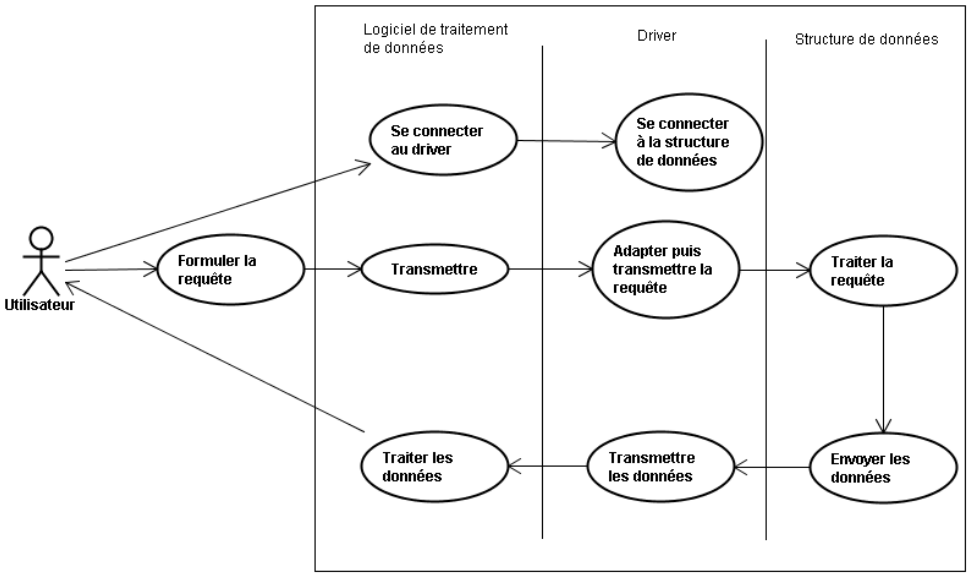


FIG. 1.2 – Diagramme de cas d'utilisation

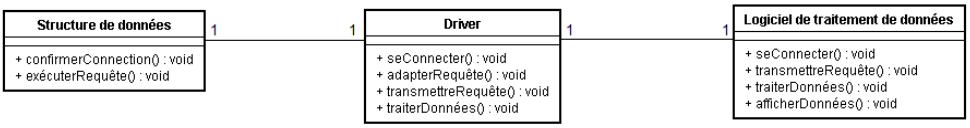


FIG. 1.3 – Esquisse du diagramme de classes

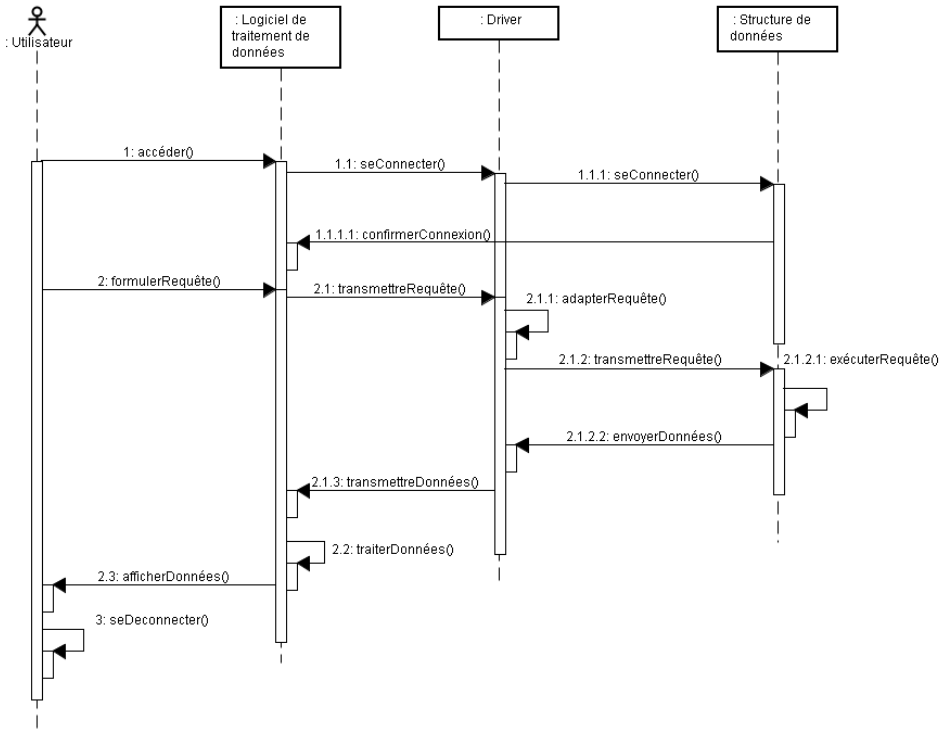


FIG. 1.4 – Diagramme de séquences

A ce stade du projet, ils ne présentent que les aspects « métiers » de l'application.

- Le diagramme de cas permet de suivre les différentes utilisations qui ont lieu lorsque l'utilisateur demande à importer des données depuis la structure de données.
- Le diagramme de classes met en évidence les différentes opérations effectuées par chaque logiciel.
- Le diagramme de séquences décrit de manière plus détaillée les interactions entre chaque entité, et donne une notion des périodes durant lesquelles les logiciels sont actifs dans le processus.

1.3.3 Exigences fonctionnelles et techniques

Documentation sur les drivers

Avant de commencer la recherche du driver, nous commencerons par consistuer une documentation suffisante pour comprendre le fonctionnement d'un driver.

Cette documentation contiendra des informations précises sur les types de driver pouvant être utilisés dans notre projet. Nous étudierons plus en détail le type de driver adopté pour la suite du projet, et en exposerons le fonctionnement et l'architecture.

Recherche de driver

Pour ne pas avoir à réécrire complètement un driver, nous partirons d'un projet libre de droit déjà existant. Nous orienterons nos recherches sur les communautés libres existantes sur Internet telles que *SourceForge*.

Le driver sera sélectionné selon les critères suivants :

- Le projet doit se rapprocher au maximum du nôtre.
- Le driver doit être capable de se connecter aux SGBD les plus classiques.
- Le driver se connecte d'une part à un outil de requête, et d'autre part à un logiciel d'où sont lues les données.
- Le driver doit être facilement analysable. Les données lues sont sous un format simple et le code est clairement commenté.
- La communauté suivant ce projet est active.

1.4 Besoins non fonctionnels

Les exigences non fonctionnelles sont les propriétés comportementales que les fonctionnalités doivent avoir, comme la performance, la facilité d'utilisation, ...

Les exigences de cette partie concernent donc l'apparence du produit, et la perception par ses utilisateurs potentiels.

On peut assigner une mesure spécifique à chaque exigence non fonctionnelle.

1.4.1 Ergonomie et convivialité du produit

L'ergonomie de notre driver n'entre ici pas en jeu, étant donné que le driver que nous devons développer n'apparaît jamais en premier plan, mais s'occupe uniquement du traitement des requêtes et des données.

Le logiciel interfaçant la communication avec l'utilisateur est donc la partie du projet qui sera concernée par la mise en place d'une ergonomie d'une convivialité à donner au produit.

Si pour une raison quelquonque l'utilisateur doit avoir à paramétrer le driver, l'interface que nous mettrons en place sera la plus simple et la plus guidée possible.

1.4.2 Facilité d'utilisation et facteurs humains

Cette partie rassemble les exigences dues aux caractéristiques des utilisateurs directs du produit.

Ici aussi, le driver n'étant jamais en communication avec l'utilisateur, certains facteurs n'entreront pas en jeu, tels que la facilité d'utilisation, la facilité d'apprentissage, la facilité de compréhension et politesse et les exigences d'accessibilité.

On peut également citer les personnalisation et internationalisation, qui n'interviendront pas en jeu dans notre driver, celui-ci ne servant qu'à transmettre des requêtes et des données. Ce sera le logiciel parent qui aura la charge de gérer les langues, à usage de l'utilisateur final.

1.4.3 Fonctionnement du produit

Cette partie permet de discuter des exigences et des résultats concernant le fonctionnement externe du produit.

Rapidité d'exécution et temps de latence

La rapidité de traitement des données n'est pas une caractéristique primordiale du driver.

Celui-ci pouvant être amené à traiter une grande capacité d'information, il serait tout de même appréciable de chercher à avoir de bonnes performances.

Précision et exactitude

On cherche à quantifier l'exactitude désirée pour le travail fourni par le produit.

Les données, étant stockées d'une manière peu complexe mais exacte (il y a peu d'erreurs sur ces données), il est très important que celles-ci ne soient pas modifiées (voire détruites) durant leur traitement par le driver.

Robustesse ou tolérance à un emploi erroné

Les circonstances anormales de fonctionnement peuvent quasiment être considérées comme « normales ». Les produits ont une ampleur et une complexité telles qu'il y a de fortes chances qu'à un moment ou un autre, un composant ne fonctionnera pas correctement.

Les exigences de robustesse servent à éviter un panne complète du produit. La robustesse spécifie l'aptitude du produit à fonctionner quand il est soumis à des circonstances anormales de fonctionnement.

Notre logiciel ne traitera pas les cas particuliers de panne ou de dysfonctionnement des logiciels avec lesquels il collabore. Nous tiendrons cependant à informer l'utilisateur, dans la mesure du possible, des opérations s'étant executées anormalement, afin que celui-ci puisse corriger l'erreur et relancer l'exécution.

Capacité de stockage et montée en charge

Cette section spécifie les volumes que le produit doit être capable de traiter et la quantité d'information qu'il doit pouvoir stocker.

Nous n'aurons pas à gérer la capacité des données que pourrait traiter notre driver. Nous essayerons tout de même d'en fixer les limites et de protéger le driver des dépassements de capacité.

Le driver sert d'intermédiaire entre les logiciel mais ne stocke pas d'information à long terme. Il faudra cependant s'assurer que le produit est capable de traiter les volumes attendus.

1.4.4 Maintenance du produit

Cette partie permet de spécifier les contraintes pour faire sur le produit des modifications. Il faut ainsi rendre chacun attentif aux besoins de maintenance du produit.

On pose donc que le système doit pouvoir être maintenu par des développeurs qui ne sont pas les développeurs d'origine.

1.4.5 Calcul des coûts

Dans le cas de ce projet, un calcul des coûts ne peut pas être effectué car cette estimation se base notamment sur des fonctions qui sont inconnues à ce stade d'avancement du projet.

1.4.6 Planning prévisionnel

On établit ici un calendrier de travail, basé sur les délais qui nous été donnés au début du projet. Ce projet se déroule du 1er octobre au 30 juin, à raison de deux demi-journées par semaine par groupe de deux étudiants.

Le projet comporte trois étapes. Chacune de ces étapes fait l'objet d'un rapport écrit et d'une soutenance orale.

Bibliographie et analyse des besoins

- Des fiches de suivi de projet seront remplies chaque semaine.
- Une note de 4-5 pages sera remise aux responsables avant le 26 octobre 2007.
- Le rapport écrit sera remis le 7 décembre 2007.
- La soutenance orale aura lieu le 18 décembre 2007.

Modélisation

- Le rapport écrit sera remis le 15 février 2008.
- La soutenance aura lieu du 18 au 29 février 2008.

Réalisation

- Le rapport écrit sera remis le 23 mai 2008.
- La soutenance aura lieu du 26 au 29 mai 2008.

1.5 Autres aspects du projet

Ces autres aspects définissent les conditions dans lesquelles le projet sera fait. Nous les incluons dans le cahier des charges afin de présenter une image cohérente des facteurs de succès ou d'échec du projet.

1.5.1 Tâches à faire pour livrer le système

La livraison du driver se fera directement, après avoir compilé ce dernier dans le format souhaité.

1.5.2 Contrôle final de qualité sur le site

Lors de la livraison du driver, on testera la complète inclusion de ce dernier avec les logiciels spécifiés dans la partie « Contraintes non fonctionnelles », logiciels qui devront collaborer avec ce driver.

1.5.3 Manuel utilisateur et formations

Un manuel utilisateur devra être fourni avec le driver, afin de pouvoir comprendre les actions de ce dernier et ses interactions avec son environnement, ainsi que expliquer comment l'intégrer à un projet.

Récapitulatif du cahier des charges

Le cahier des charges nous a permis de discerner les différentes tâches à accomplir, ainsi que les problèmes qu'il nous faudra résoudre dans la suite du projet. Il a permis de poser un cadre auquel devra se conformer le résultat final de notre projet.

D'autre part, la réalisation de ce cahier des charges a été l'occasion pour nous de découvrir comment rédiger un cahier des charges.

Chapitre 2

Etude bibliographique du domaine

Ce chapitre va permettre de faire la synthèse de la bibliographie, en faisant l'utilisation des renvois bibliographiques dans le corps du rapport.

2.1 Documentation sur les drivers

Afin d'assurer la communication entre l'outil de requête et le logiciel de M. Guérin, nous utiliserons un driver.

Cette documentation a pour rôle de définir ce qu'est un driver, et de nous aider à déterminer le type de driver que nous utiliserons dans la suite de notre projet.

2.1.1 Qu'est ce qu'un driver ?

Avant de s'intéresser aux différents types de driver, nous allons définir ce qu'est un driver, et expliquer comment il fonctionne.

Le cas général du driver matériel

Généralement, un driver est un exécutable permettant à un logiciel haut niveau de communiquer avec un périphérique matériel [1].

Lorsque le logiciel demande au matériel d'effectuer une tâche, le driver traduit la demande, formulée en langage « haut niveau », en commandes « bas niveau » interprétables par le matériel. De même, les résultats de la tâche renvoyés par le matériel sont traduits par le driver avant d'être retransmis au logiciel.

Les drivers haut niveau

Si la définition ci-dessus est correcte pour la grande majorité des drivers, elle ne s'applique pas aux quelques drivers qui ne font pas appel à un périphérique matériel.

Il est possible de demander à un driver d'effectuer des opérations « haut niveau ». C'est le cas des drivers imprimantes utilisant le *Postscript*, qui modifient les données avant de les traduire et de les envoyer au matériel.

Composés uniquement de fonctions haut niveau, certains drivers font complètement abstraction de la partie matérielle de la définition-ci-dessus. On pourra citer le célèbre *PDFCreator* qui, en se faisant passer pour un driver imprimante, convertit les fichiers *Word* au format *PDF*.

C'est également vrai pour les drivers permettant de se connecter à des bases de données tels que les drivers ODBC. Ceux-ci se connectent d'une part sur un SGBD ou un logiciel de traitement de données, et d'autre part sur un logiciel utilisant une base de donnée ou directement sur un fichier de données.

Leur rôle est cependant le même que celui des drivers matériels : il sert de traducteur entre les deux éléments à connecter.

Comment fonctionne un driver ?

Pris en charge par les systèmes d'exploitation, les drivers matériels sont faciles à installer.

A son démarrage, le système d'exploitation référence tous les nouveaux périphériques connectés à l'ordinateur. Un programme d'installation fourni par le constructeur du matériel permet alors de lier cette référence avec le driver approprié. Le système d'exploitation garde également en mémoire la localisation de chaque driver.

Par la suite, à chaque fois qu'un logiciel aura besoin d'utiliser ce matériel, le système d'exploitation lui indiquera le driver associé à lancer pour établir la communication.

Pour un driver permettant de se connecter à une base de données, la manipulation d'installation se complique un peu.

Le système permettant de lier un driver à un logiciel utilisant une base de données n'existe pas forcément. Pour ne pas arranger les choses, différents types de drivers peuvent être utilisés pour réaliser une connexion sur un même logiciel.

Il est souvent nécessaire d'indiquer manuellement à chaque logiciel quel est le type du driver que l'on souhaite utiliser, où se situe celui-ci et à quel logiciel le driver doit se connecter [2]. Une fois connectés, les logiciels peuvent communiquer entre eux de la même façon qu'avec un pilote matériel.

2.1.2 Les différents types de drivers établissant une connexion avec un SGBD

Il existe différents types de drivers permettant la communication entre des clients de bases de données et les SGBD du marché. Ces drivers se distinguent par leurs architectures, leurs protocoles, leurs interfaces, leurs performances, leurs caractéristiques d'implémentation ou encore par le nombre de logiciels auxquels ils sont capables de se connecter.

Pour réaliser notre projet, nous avons du choisir celui qui nous a paru le plus adapté parmis la liste des drivers les plus courants : *ODBC*, *JDBC*, *OLE DB*, *ADO*, ...

Voici une brève présentation des protocoles et la justification de notre choix.

Les drivers ODBC

Le type de driver Open Data Base Connectivity (ODBC) est le premier sur lequel nous nous sommes renseignés.

Ce type de driver développé par Microsoft a l'avantage d'être un standard parmi les drivers. Étant le plus ancien et le plus populaire des types présentés ici, la quasi-totalité des SGBD et des logiciels traitant des bases de données sont compatibles avec ce type de driver.

Nous avons cependant dû rejeter ce type de driver car celui-ci est développé en C, et non en Java tel qu'exigé dans notre projet. Une autre raison qui nous a fait écarter ce driver est le fait qu'un autre type de driver, les drivers JDBC, jouit d'une popularité bien plus importante dans la communauté du libre, dans laquelle nous comptons trouver un driver Open Source.

Les drivers OLE DB et ADO

OLE DB et ADO sont des protocoles de communication entre application développés par Microsoft dans le but de remplacer ODBC. Nous ne nous sommes pas aussi longuement attardés sur ces deux types de drivers.

Si ceux-ci sont plus récents et plus performants que ODBC, leur manque de popularité leur fait défaut. Ils ne réunissent pas encore une communauté suffisante pour que l'on puisse trouver une bonne documentation et un projet libre existant répondant à nos exigences.

Les drivers JDBC

De part sa prédestination au langage Java, le protocole JDBC [3] [4] nous parait être le plus intéressant.

Très similaire au type ODBC, le type de driver JDBC est également capable de se connecter à la grande majorité des SGBD du marché [5]. Certains drivers JDBC sont également capables d'établir un pont avec le protocole ODBC pour atteindre l'ensemble des SGBD compatibles avec ce dernier [6].

Les avantages du type de driver JDBC par rapport à ODBC sont les suivants :

- les drivers JDBC sont nativement codés en Java, ce qui les rend particulièrement portables.
- L'engouement de la communauté libre pour le protocole JDBC est bien plus important que pour ODBC. Sur SourceForge.net, on compte 548 projets libres recensés en relation avec le protocole JDBC, contre 176 pour ODBC [7].

On notera tout de même que les drivers JDBC ont le désavantage d'être moins bien pris en charge par Windows que les drivers ODBC. Malgré cela, et partant du principe que la recherche d'un driver libre est une étape critique pour l'avancée de notre projet, nous continuerons notre projet en travaillant sur le protocole JDBC.

2.1.3 Les drivers JDBC

En continuant nos recherches sur les drivers utilisant le protocole JDBC, développé par SUN [8], nous avons découvert que ceux-ci se décomposent en quatre familles distinctes disposant d'architectures différentes [6].

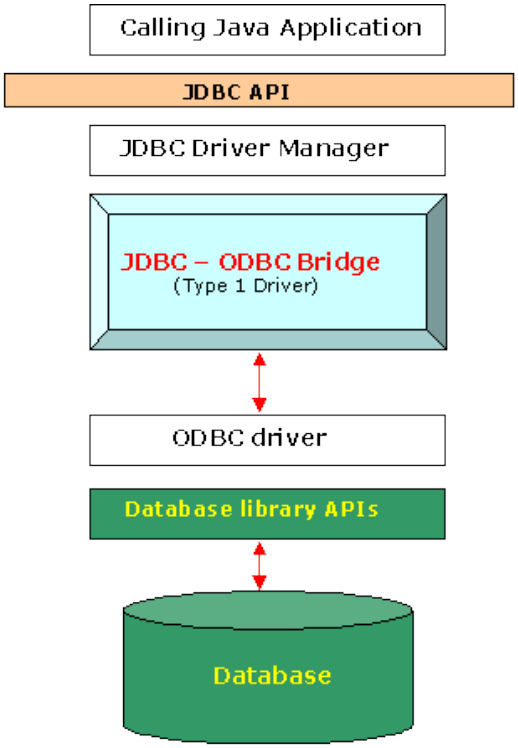


FIG. 2.1 – Schéma d'un driver JDBC de type 1 [9]

Pont JDBC-ODBC (Type 1 - FIG 2.1)

Ce driver accède à la base de données en passant par l'intermédiaire du protocole ODBC.

Les appels formulés par le logiciel de requête utilisant le protocole JDBC sont alors convertis en appels ODBC, avant d'être envoyés au SGBD (ici, la structure de données) dont on veut exporter les données. De même, les données renvoyées par le SGBD doivent être traduites avec le protocole ODBC puis JDBC avant d'être exploitables.

Comme on peut l'imaginer, ce protocole est particulièrement lent. En revanche, il possède l'avantage de pouvoir accéder à tous les SGBD compatibles avec ODBC.

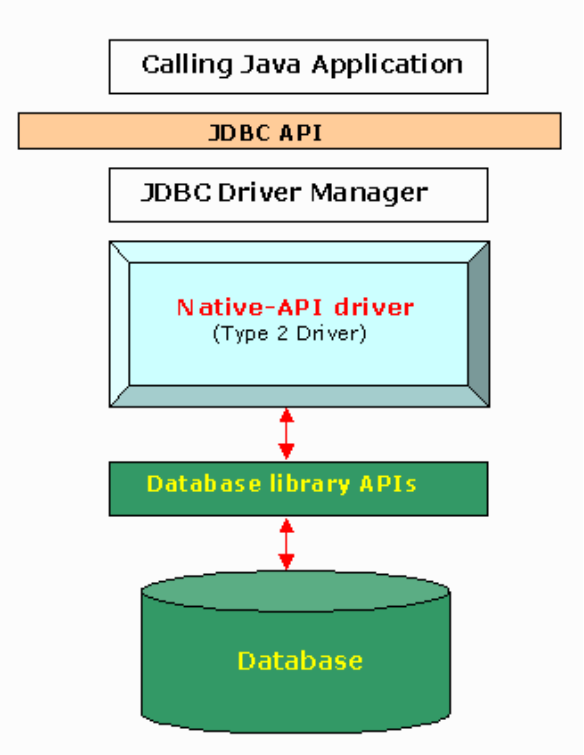


FIG. 2.2 – Schéma d'un driver JDBC de type 2 [9]

Driver d'API natifs (Type 2 - FIG 2.2)

Ce type de driver ne demande pas au SGBD sur lequel on veut se connecter d'implémenter une interface complexe. Il convertit les demandes formulées par l'outil de requêtes en appels natifs pour l'API du SGBD généralement en C ou en C++.

Ce driver est plus rapide que le driver de type 1. Malgré cela, pour réaliser la translation, le driver nécessite d'être codé à la fois en Java et en C, ce qui va à l'encontre des exigences de notre projet. De plus, il faut que le SGBD sur lequel on se connecte dispose d'une bibliothèque d'API natives, ce qui n'est pas le cas de toutes les SGBD.

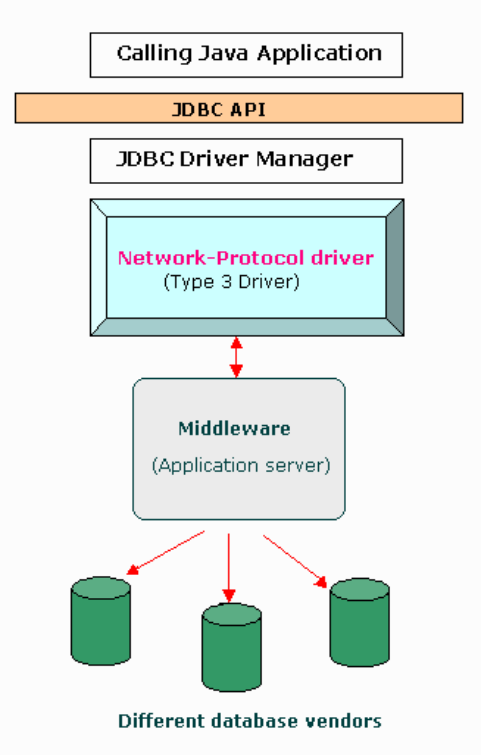


FIG. 2.3 – Schéma d'un driver JDBC de type 3 [9]

Drivers utilisant un middleware (Type 3 - FIG 2.3)

Ce driver s'appuie sur un logiciel intermédiaire (*middleware*), placé entre le driver et une ou plusieurs bases de données. Ce logiciel intermédiaire, généralement serveur de l'application, permet à la fois de traduire les appels en JDBC pour le SGBD, de gérer les communications à distance et d'assurer la sécurité des échanges.

Cette solution, difficile à mettre en place, nécessite en plus du développement du driver, le développement du middleware. Cependant, cette solution reste très intéressante. Aussi, le driver purement développé en JAVA est très portable.

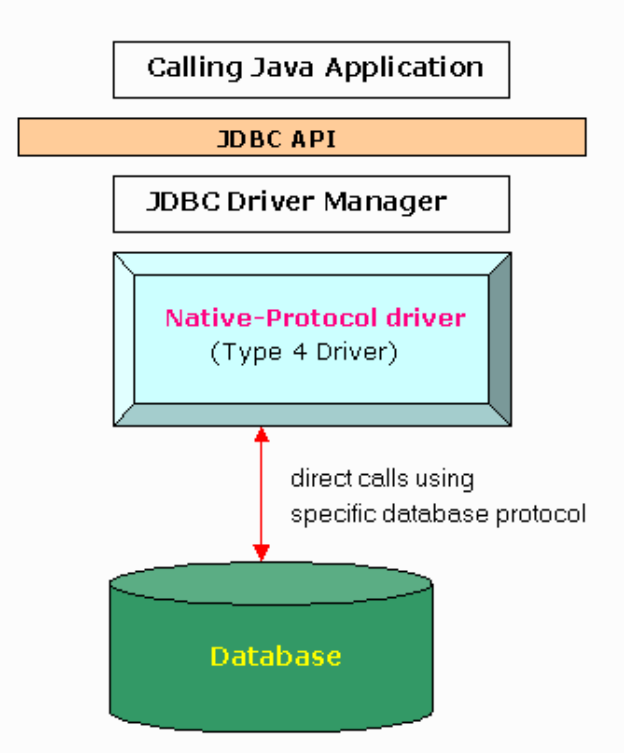


FIG. 2.4 – Schéma d'un driver JDBC de type 4 [9]

Drivers utilisant un protocole réseau natif (Type 4 - FIG 2.4)

Le driver traduit les requêtes JDBC en un protocole réseau directement interprétable par le SGBD.

Cette solution parfaitement portable est la plus rapide de toutes et la plus adaptée à une base de données accessible par Intranet. La majorité des SGBD dispose d'un pareil protocole réseau.

Pour développer un pareil driver, il faut encapsuler l'interface cliente du SGBD dans le driver. Le logiciel que notre driver viendra compléter pouvant s'adapter au driver, cela ne posera pas de problème. Alliant simplicité, rapidité et portabilité, le driver de type 4 est donc très adapté à notre projet.

Récapitulatif sur les drivers

La réalisation de ce document nous a permis de mieux comprendre les caractéristiques et le fonctionnement des drivers.

Si son but était de nous aider à déterminer le type de driver que nous choisirons pour la suite du projet, nous nous sommes aperçus tardivement que le driver JDBC ne répond peut-être pas à nos attentes. En effet, si les drivers JDBC sont capables de se connecter à un très grand nombre de bases de données et d'être utilisés par des logiciels de traitement de requêtes tels qu'Open Office, nous n'avons pas réussi à utiliser ce type de driver avec le module d'importation de données d'Excel.

Après quelques recherches, nous avons compris que le driver JDBC, bien que capable de se connecter à un grand nombre de structures de données, est surtout utilisé dans le dévelopement de logiciels nécessitant d'importer des données. Il semble difficile, sinon impossible, de l'utiliser pour importer des données vers un grand nombre de logiciels déjà existants tels qu'Excel et Access.

Nous continuerons les recherches dans cette voie si nos doutes se confirment. Nous passerons surement à la recherche d'un driver de type ODBC. Ce driver devra alors être codé en langage C.

2.2 Licences libres

Une licence de logiciel libre [10] est une licence s'appliquant à un logiciel pour en faire un contenu libre offrant à l'utilisateur certains droits quant à l'utilisation, à la modification, à la rediffusion et à la réutilisation de ce logiciel dans des logiciels dérivés [11].

Une licence de logiciel libre applique quatre types de liberté pour l'utilisateur d'un logiciel :

- La liberté d'exécuter le programme, pour tous les usages
- La liberté d'étudier le fonctionnement du programme (ceci suppose l'accès au code source)
- La liberté de redistribuer des copies (ceci comprend la liberté de vendre des copies)
- La liberté d'améliorer le programme et de publier ses améliorations (ceci suppose l'accès au code source)

Les logiciels libres sont souvent divisés en trois [12], selon le degré de liberté accordé par la licence en matière de redistribution [13].

2.2.1 Domaine public

En langage courant : le logiciel appartient à tout le monde. C'est une caractéristique juridique qui n'a pas besoin de licence du fait que le logiciel n'a aucun ayant-droit. Théoriquement, tout logiciel tombe dans le domaine public une fois les droits d'auteur échus.

2.2.2 Licences protectrices (Copyleft)

Il s'agit des licences qui interdisent la redistribution hors des principes du copyleft, car « Si un programme est un logiciel libre au moment où il quitte les mains de son auteur, cela ne signifie pas nécessairement qu'il sera un logiciel libre pour quiconque en possédera une copie ».

Les licences du projet GNU sont les plus célèbres. Une telle licence permet d'intégrer du logiciel sous licence BSD et de le redistribuer sous licence GPL. L'inverse est impossible.

Des acteurs des projets BSD critiquent un degré de liberté moindre des licences de type copyleft, et des acteurs commerciaux dénoncent une nature contaminante.

Voici une liste des principales licences libres non adaptées au projet (virales) :

GPL (« GNU General Public License »)

La GPL [14] ne donne le droit de redistribuer un logiciel [15] que si l'ensemble du logiciel, y compris toutes les éventuelles modifications, sont redistribuées selon les termes exacts de la GPL [16].

Cette licence est dite « virale » ou « contaminante » par ses opposants, car si elle autorise la fusion d'un logiciel sous GPL avec un logiciel sous une autre licence, elle n'autorise en revanche la redistribution du logiciel fusionné que sous GPL.

Les termes de la GPL autorisent toute personne à recevoir une copie d'un travail sous GPL. La GPL ne donne pas à l'utilisateur des droits de redistribution sans limite. Le droit de redistribuer est garanti seulement si l'utilisateur fournit le code source de la version modifiée. En outre, les copies distribuées, incluant les modifications, doivent être aussi sous les termes de la GPL.

LGPL (« GNU Lesser General Public License »)

Cette licence limitée, ou amoindrie [17], a été créée pour permettre à certains logiciels libres de pénétrer tout de même certains domaines où le choix d'une publication entièrement libre de toute l'offre était impossible.

LGPL est une version modifiée de GPL qui a l'avantage d'être moins contraignante quant à son utilisation dans un contexte de cohabitation avec des logiciels propriétaires. Cette licence permet de s'affranchir du caractère héréditaire de la licence GPL.

Ainsi, il devient possible à un programmeur désireux de faire un logiciel propriétaire, d'utiliser certains outils du monde libre (ex : la bibliothèque graphique GTK) sans contraindre son logiciel à l'être également. Cependant, toute modification de code source dans la bibliothèque LGPL devra également être publiée sous la licence LGPL.

2.2.3 Licences non protectrices

Il s'agit des licences qui offrent la plus grande liberté. En général, seule la citation des auteurs originaux est demandée. En particulier, ces licences permettent de redistribuer un logiciel libre sous une forme non libre. Ces licences permettent donc à tout acteur de changer la licence sous laquelle le logiciel est distribué.

Un cas de changement de licence courant est l'intégration de logiciel sous licence BSD dans un logiciel sous copyleft (licence GPL). Un autre cas courant est l'intégration de logiciel sous licence BSD dans les logiciels propriétaires.

Voici une liste des principales licences libres adaptées au projet (non-virales), parmi lesquelles une que nous choisirons dans le cadre de notre projet :

BSD License (Berkeley software distribution license)

La licence BSD est une licence libre utilisée pour la distribution de logiciels.

Elle permet de réutiliser tout ou partie du logiciel sans restriction, qu'il soit intégré dans un logiciel libre ou propriétaire. La licence BSD permet l'utilisation commerciale dans des produits propriétaires. Les logiciels publiés selon les termes de cette licence peuvent être incorporés dans des solutions propriétaires ou/et commerciales.

Les travaux basés sur des éléments souslicence BSD peuvent même être publiés sous une licence propriétaire (qui doit toutefois respecter les clauses mentionnées dans la licence BSD). Parmi les exemples notoires, on peut citer l'utilisation par Sun de code réseau sous BSD, et des composants en provenance de FreeBSD dans Mac OS X.

Le code sous licence BSD peut être publié sous licence GPL sans le consentement des auteurs originaux puisque les termes de la GPL respectent tous ceux de la licence BSD. Par contre, du code sous licence GPL ne peut pas être mis sous licence BSD sans l'autorisation des auteurs car la licence BSD ne respecte pas toutes les contraintes imposées par la licence GPL. En publiant du code GPL sous licence BSD, on autoriserait par exemple la redistribution sans fournir le code source alors que c'est interdit par les termes de la licence GPL.

Voici une partie du texte de la licence :

```
Copyright (c) <YEAR>, <OWNER>
All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the following
conditions are met:
  * Redistributions of source code must retain the above
  copyright notice, this list of conditions and the following
  disclaimer.
  * Redistributions in binary form must reproduce the above
  copyright notice, this list of conditions and the following
  disclaimer in the documentation and/or other materials
  provided with the distribution.
  * Neither the name of the <ORGANIZATION> nor the names of
  its contributors may be used to endorse or promote products
  derived from this software without specific prior written
  permission.
```

Licence Apache (ASL)

Le point majeur de la licence Apache [18] est d'une part d'autoriser la modification et la distribution du code sous toute forme (libre et commerciale) et d'autre part d'obliger le maintien du copyright lors de toute modification (et également du texte de la licence elle-même).

Voici le texte de la licence :

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on
an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
either express or implied.
See the License for the specific language governing permissions
and limitations under the License.
```

Licence MIT (X Window System)

C'est une licence de logiciel libre et Open Source.

Elle donne à toute personne recevant le logiciel le droit illimité de l'utiliser, le copier, le modifier, le fusionner, le publier, le distribuer, le vendre et de changer sa licence. La seule obligation est de mettre le nom des auteurs avec la notice de copyright.

Elle est très proche de la nouvelle licence BSD, seule la dernière clause diffère. Elle est compatible avec la GNU General Public License.

Voici le texte de la licence :

```
Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom
the Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.
```

Récapitulatif sur les licences

Dans le cadre de notre projet, et selon les exigences fournies, nous choisirons donc l'une des licnces non-virales citées ci-dessus.

2.3 Sites de dépôt de projet

Afin de pouvoir distribuer facilement le projet réalisé, nous devrons nous appuyer sur des sites de partages de projet. A ce jour, deux sites représentent la majorité des activités de partage de projet : SourceForge et FreshMeat.

Nous allons donner une description de ces deux sites afin de mieux les connaître, et ainsi de pouvoir choisir celui qui sera le plus à même de recevoir notre projet.

2.3.1 SourceForge

Description

Sourceforge, produit par VA Software et lancé en 1999, est un système de gestion de développement de logiciel collectif et de gestion de version. Il fournit une interface unifiée à une série de logiciels serveur et intègre plusieurs applications à source ouverte et/ou libre (notamment CVS).

SourceForge.net est un site web hébergeant la gestion du développement de logiciels à source ouverte, lui-même hébergé par VA Software qui utilise une version du logiciel SourceForge. Environ 160 000 projets sont hébergés sur le site, beaucoup d'entre eux étant dormants ou n'impliquant qu'un seul développeur. Actuellement, environ 1,7 millions d'utilisateurs sont enregistrés.

Caractéristiques

Ce site offre un accès simple et libre d'hébergement et d'outils pour les développeurs de projets libres / Open Source depuis plusieurs années, et s'est rapidement fait connaître dans les communautés de développement informatique pour tous ses services.

SourceForge.net autorise n'importe quel projet, à condition que celui-ci soit enregistré dans un sous-domaine type (tel que http ://project-name.sf.net ou http ://project-name.sourceforge.net). Cette adresse de diffusion donnée

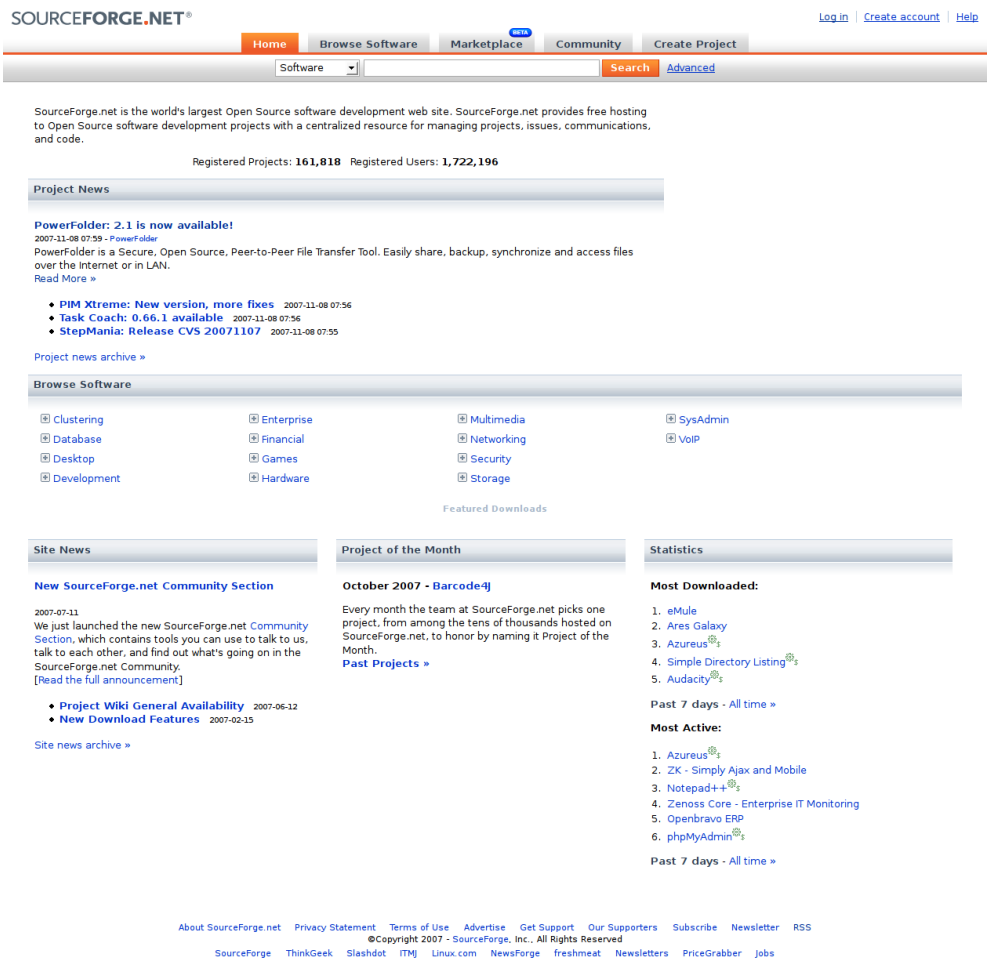

SOURCEFORGE.NET
Log in | Create account | Help
Home
Browse Software
Marketplace
BETA
Community
Create Project
Software
Search
Advanced
SourceForge.net is the world's largest Open Source software development web site. SourceForge.net provides free hosting to Open Source software development projects with a centralized resource for managing projects, issues, communications, and code.
Registered Projects: 161,818 Registered Users: 1,722,196
Project News
PowerFolder: 2.1 is now available!
2007-11-08 07:59 - PowerFolder
PowerFolder is a Secure, Open Source, Peer-to-Peer File Transfer Tool. Easily share, backup, synchronize and access files over the Internet or in LAN.
Read More »
PIM Xtreme: New version, more fixes 2007-11-08 07:56
Task Coach: 0.66.1 available 2007-11-08 07:56
StepMania: Release CVS 20071107 2007-11-08 07:55
Project news archive »
Browse Software
Clustering
Database
Desktop
Development
Enterprise
Financial
Games
Hardware
Multimedia
Networking
Security
Storage
SysAdmin
VoIP
Featured Downloads
Site News
New SourceForge.net Community Section
2007-07-11
We just launched the new SourceForge.net Community Section, which contains tools you can use to talk to us, talk to each other, and find out what's going on in the SourceForge.net Community.
[Read the full announcement]
Project Wiki General Availability 2007-06-12
New Download Features 2007-02-15
Site news archive »
Project of the Month
October 2007 - Barcode4J
Every month the team at SourceForge.net picks one project, from among the tens of thousands hosted on SourceForge.net, to honor by naming it Project of the Month.
Past Projects »
Statistics
Most Downloaded:
1. eMule
2. Ares Galaxy
3. Azureus
4. Simple Directory Listing
5. Audacity
Past 7 days - All time »
Most Active:
1. Azureus
2. ZK - Simply Ajax and Mobile
3. Notepad++
4. Zenoss Core - Enterprise IT Monitoring
5. Openbravo ERP
6. phpMyAdmin
Past 7 days - All time »
About SourceForge.net Privacy Statement Terms of Use Advertise Get Support Our Supporters Subscribe Newsletter RSS
©Copyright 2007 - SourceForge, Inc., All Rights Reserved
SourceForge ThinkGeek Slashdot ITMJ Linux.com NewsForge freshmeat Newsletters PriceGrabber Jobs

FIG. 2.5 – Page d'accueil de SourceForge.net

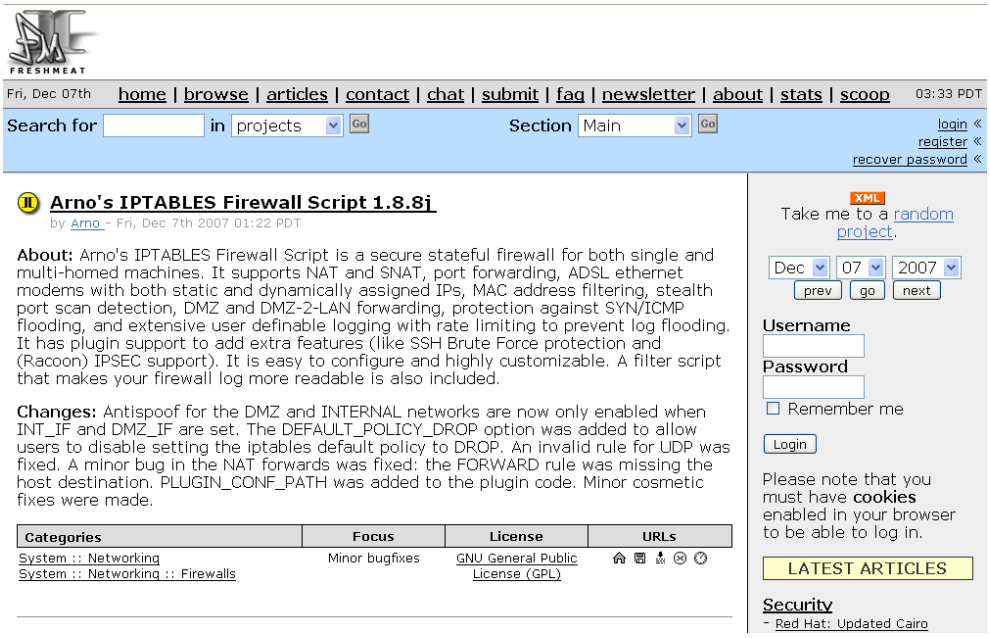

FRESHMEAT
Fri, Dec 07th
home | browse | articles | contact | chat | submit | faq | newsletter | about | stats | scoop
03:33 PDT
Search for
in projects
Go
Section Main
Go
login
register
recover password
Arno's IPTABLES Firewall Script 1.8.8j
by Arno - Fri, Dec 7th 2007 01:22 PDT
About: Arno's IPTABLES Firewall Script is a secure stateful firewall for both single and multi-homed machines. It supports NAT and SNAT, port forwarding, ADSL ethernet modems with both static and dynamically assigned IPs, MAC address filtering, stealth port scan detection, DMZ and DMZ-2-LAN forwarding, protection against SYN/ICMP flooding, and extensive user definable logging with rate limiting to prevent log flooding. It has plugin support to add extra features (like SSH Brute Force protection and (Racoon) IPSEC support). It is easy to configure and highly customizable. A filter script that makes your firewall log more readable is also included.
Changes: Antispoof for the DMZ and INTERNAL networks are now only enabled when INT_IF and DMZ_IF are set. The DEFAULT_POLICY_DROP option was added to allow users to disable setting the iptables default policy to DROP. An invalid rule for UDP was fixed. A minor bug in the NAT forwards was fixed: the FORWARD rule was missing the host destination. PLUGIN_CONF_PATH was added to the plugin code. Minor cosmetic fixes were made.
Categories
Focus
License
URLs
System :: Networking
System :: Networking :: Firewalls
Minor bugfixes
GNU General Public License (GPL)
XML
Take me to a random project.
Dec
07
2007
prev
go
next
Username
Password
Remember me
Login
Please note that you must have cookies enabled in your browser to be able to log in.
LATEST ARTICLES
Security
- Red Hat: Updated Cairo

FIG. 2.6 – Page d'accueil de FreshMeat

pour chaque projet, rend l'acessibilité assez aisée et permet une consultation rapide des projets en cours de réalisation. Grâce à un nombre élevé d'utilisateurs enregistrés (plus d'un million) et une recherche réellement simple, n'importe quel projet peut facilement être consulté.

La communauté active des développeurs permet d'obtenir des réponses rapides aux questions qui peuvent se poser. SourceForge.net supporte un large espace de stockage et permet la mise en place de multiples fonctioinnalités, telles qu'un wiki, une base de données MySQL, une gestion des versions de code source (avec CVS ou SVN), et permet même la diffusion du site web du développeur au nom de domaine réservé.

2.3.2 FreshMeat

Description

Freshmeat est un site web répertoriant un grand nombre de logiciels, majoritairement libres.

Il permet aussi de suivre leur évolution, d'écrire ou de lire des critiques de ces logiciels ou des articles, de dialoguer avec les auteurs de logiciels, ...

Bien que la majorité des logiciels du sites soient des logiciels libres ou Open Source pour les systèmes Unix ou Linux, on y trouve aussi des logiciels pour d'autres OS comme Windows, ainsi que des logiciels fermés ou commerciaux.

Freshmeat appartient à l'Open Source Technology Group (OSTG) dirigé par VA Software.

Caractéristiques

Les programmeurs ont la possibilité d'inscrire leurs projets et d'indiquer l'arrivée des mise-à-jour tandis que les utilisateurs ont la possibilité de les télécharger et (parfois) de les noter ou de laisser des commentaires.

Une recherche des logiciels est possible, selon plusieurs critères : domaine, licence, état du projet, environnement, public visé, système d'exploitations pris en charge, langage de programmation employé et langues disponibles.

La totalité de la base de données de Freshmeat concernant les réalisations logicielles est librement téléchargeable.

2.3.3 Autres sites

On peut remarquer que d'autres sites de partage de projet informatiques existent à ce jour, tels que RubyForge, Tigris.org, BountySource, BerliOS, JavaForge and GNU Savannah. Mais ces moyens de partage restent assez peu utilisés dans le monde du développement.

Etant donné que pour le développement de notre driver, nous avons besoin d'une importante communauté permettant de nous aider dans notre travail, nous ne continuerons pas à nous intéresser à ces différents sites.

Récapitulatif sur les sites de dépôt

On remarque que les deux sites SourceForge et FreshMeat sont très utilisés pour le partage de projets informatiques. Ces deux moyens de gestion de projet en ligne sont très appréciés, notamment en raison de leur gratuité ; ils sont ainsi accessibles à tous. Ils permettent de facilement réaliser des projets libres, ceux-ci étant faits avec des équipes de développeurs formés sur internet.

Ce système de gestion des versions utilisé, CVS, est très intéressant car il permet d'accéder aux données partagées, et ceci depuis n'importe quel PC. Les projets sont ainsi facilement modifables, et permet une gestion très simple de l'évolution des versions du projet au cours du temps.

2.4 Gestion des versions

Afin de pouvoir distribuer facilement le projet réalisé, nous avons étudié les différents sites de partages de projet.

Mais lorsque plusieurs personnes travaillent en collaboration, et parfois en même temps, sur un même projet, il est important de pouvoir gérer les conflits entre les modifications du projet, et les différentes versions du logiciel en découlant.

Afin de pouvoir gérer ces nombreux conflits, nous allons devoir utiliser un logiciel de gestion de version. Ces logiciels ou utilitaires gèrent les mises à jour conflictuelles en évitant le risque de perdre des données.
Nous allons détailler le principe de deux principaux systèmes de gestion de version : CVS et SVN.

2.4.1 CVS

Nous allons décrire le système de gestion de versions CVS [19].

Description

CVS, acronyme de Concurrent Versions System, est un système de gestion de versions libre, successeur de SCCS, originellement écrit par Dick Grune en 1986, puis complété par Brian Berliner (avec le programme CVS lui-même) en 1989, et par la suite amélioré par de très nombreux contributeurs.

CVS est devenu populaire dans le monde du logiciel Open Source, et est réalisé sous la licence *GNU General Public Licence*. Bien qu'il soit toujours

très utilisé dans le domaine du logiciel libre, le logiciel concurrent Subversion apporte de nouvelles fonctionnalités de gestion des sources.

On notera qu'il existe aussi des logiciels décentralisés comme Bazaar, Darcs, Git, Mercurial ou Monotone, et toujours sous licence Open Source.

Parmi les interfaces graphiques les plus populaires, citons sous Linux, le programme Cervisia et sous Windows TortoiseCVS.

Caractéristiques

CVS est un système Open Source de contrôle des versions. Il permet de conserver une trace de la totalité du travail et de chaque changement dans un ensemble de fichiers. Ce système concerne notamment l'implémentation d'un projet de développement, et autorise plusieurs développeurs à collaborer.

Puisqu'il aide les sources à converger vers la même destination, on dira que CVS fait la gestion concurrente de versions ou de la gestion de versions concurrentes.

Il peut aussi bien fonctionner en mode ligne de commande qu'à travers une interface graphique. Il se compose de modules clients et d'un ou plusieurs modules serveur pour les zones d'échanges.

Principe

CVS utilise une architecture client-serveur : le serveur enregistre la ou les versions actuelles du projet et l'historique correspondant, le client se connecte au serveur afin de récupérer une copie complète du projet, travailler sur cette copie et, plus tard, sauvegarder ces changements.

Plusieurs développeurs peuvent ainsi travailler en concurrence sur le même projet, chacun éditant les fichiers de leur propre copie du projet, pour ensuite envoyer les modifications au serveur.

Pour éviter le risque que chacun se marche dessus, le serveur accepte uniquement les changements faits sur la version la plus récente du fichier. Les développeurs doivent ainsi garder leur copie du fichier à jour, en incorporant les changements des autres régulièrement.

Cette tâche est, la plupart du temps, automatiquement prise en charge par le client CVS, requiérant une intervention manuelle seulement quand un conflit est avéré entre une modification « envoyée » et une version locale d'un fichier non encore mise à jour.

Si l'opération de mise à jour réussit, alors les numéros de version de tous les fichiers impliqués s'incrémentent automatiquement, le serveur CVS ajoute une ligne de description à destination de l'utilisateur, contenant le nom de l'auteur et la date de modification, aux fichiers journal.

Fonctionnalités

Le serveur CVS peut avertir par e-mail les utilisateurs concernés par la mise à jour d'un fichier.

Les clients peuvent comparer les versions, requêter un historique complet des changements, ou obtenir une « photographie » chronologique du projet à partir d'une date ou d'un numéro de révision. Ils peuvent aussi utiliser la commande de mise à jour afin de mettre à jour leur copie locale vers la nouvelle version du serveur.

Cela élimine le besoin de répéter le téléchargement du projet en entier.

L'une des autres fonctionnalités très intéressantes de CVS est la maintenance de différentes branches dans un projet.

C'est-à-dire que pour une instance de projet donnée, une version peut former une branche séparée, utilisée pour les fichier "buggués", quand une autre version, en train d'être développée avec d'importants changements et de nouvelles fonctionnalités, forme une branche séparée.

CVS utilise la compression de donnée spour un stockage plus efficace des différentes versions d'un même fichier.

L'implémentation favorise les fichiers avec de nombreuses lignes (souvent les fichiers textes).

Limites

Il existe aussi quelques limitations à l'utilisation de CVS.

On remarque notamment que le déplacement ou renommage des fichiers ou des répertoires ne seront pas numérotés. En effet, cela va en l'encontre du principe de gestion des versions en concurrence de CVS.

Heureusement, CVS étant Open Source, les fonctionnalités de CVS évoluent très régulièrement. Ainsi, apparaissent souvent de nouvelles fonctionnalités ajoutées par les développeurs, ce qui en fait une méthode extrêment utilisée.

2.4.2 SVN

Nous allons décrire le système de gestion de versions SVN [20].

Description

Subversion (en abrégé SVN) est un système de gestion de versions, distribué sous licence Apache et BSD. Il a été conçu pour remplacer CVS.

Ses auteurs s'appuient volontairement sur les mêmes concepts (notamment sur le principe du dépôt centralisé et unique) et considèrent que le modèle de CVS est le bon, et que seule son implémentation est en cause.

Le projet a été lancé en février 2000 par CollabNet, avec l'embauche par Jim Blandy de Karl Fogel, qui travaillait déjà sur un nouveau gestionnaire de version.

Les apports de Subversion

Subversion a été écrit afin de combler certains manques de CVS.

Voici les principaux apports :

- Les *commits*, ou publications des modifications, sont atomiques.
  Un serveur Subversion utilise de façon sous-jacente une base de données capable de gérer les transactions atomiques (le plus souvent Berkeley DB)

- Subversion permet le renommage et le déplacement de fichiers ou de répertoires sans en perdre l'historique
- Les méta-données sont versionnées : on peut attacher des propriétés, comme les permissions, à un fichier, par exemple.

Du point de vue du simple utilisateur, les principaux changements lors du passage à Subversion, sont :

- Les numéros de révision sont désormais globaux (pour l'ensemble du dépôt) et non plus par fichier : chaque patch a un numéro de révision unique, quels que soient les fichiers touchés.
  Il devient simple de se souvenir d'une version particulière d'un projet, en ne retenant qu'un seul numéro.
- `svn rename` (ou `svn move`) permet de renommer (ou déplacer) un fichier
- Les répertoires et méta-données sont versionnés.

L'une des particularités de Subversion est qu'il ne fait aucune distinction entre un label, une branche et un répertoire. C'est une simple convention de nommage pour ses utilisateurs.

Il devient ainsi très facile de comparer un label et une branche ou autre croisement.

De plus, SVN dispose de très nombreues commandes supplémentaires, et beaucoup de logiciels annexes peuvent se greffer à SVN. Enfin, otre l'outil en ligne de commande, plusieurs logiciels pourvus d'interface graphique existent, afin de donner de la convivialité à SVN.

Récapitulatif des gestionnaires de versions

Actuellement, différents systèmes existent, mais seulement les deux précédemment détaillés sont réellement utilisés. La mise en place d'un tel système, dans le cadre de notre projet transversal, se fera dans les toutes prochaines semaines.

Récapitulatif de l'étude bibliographique

Après avoir rassemblé une partie des connaissances dont nous aurons besoin pour la suite de notre projet, nous serons à même de mieux comprendre

comment résoudre les problèmes auxquels nous serons confrontés.

Cependant, cette bibliographie reste très incomplète, en particulier à cause du problème du choix de driver (cf. conclusion du document sur les drivers). Nous la compléterons donc dans la suite du projet.

Récapitulatif

Malgré que l'étude bibliographique du domaine soit incomplète, nous avons, grâce à cette étude et à l'analyse des besoins, pu définir les exigences « primaires » du projet transversal.

Toutes ces définitions et ces besoins seront utiles et nécessaires pour la continuation de notre projet et l'étape à suivre.

Deuxième partie

Modélisation

Après avoir réalisé une approche bibliographique et une analyse des besoins de notre projet, nous allons désormais nous attaquer à la conception de celui-ci.

L'objectif est de proposer l'architecture générale et détaillée du logiciel en s'appuyant sur les besoins consignés dans le cahier des charges, et en modélisant le problème à l'aide de formalismes adaptés.

Afin de recentrer les notions qui seront abordées tout au long de cette phase de conception, nous rappellerons tout d'abord les termes importants qui ont été introduits dans la précédente phase de bibliographie.

Pour permettre la modélisation de notre logiciel, nous lancerons ensuite une recherche de solution pour le projet.

Enfin, nous présenterons la conception de notre projet, notamment à l'aide de schémas.

Chapitre 3

Complément de bibliographie

Afin de recentrer les notions qui seront abordées tout au long de cette phase de conception, ce chapitre nous pemettrera de rappeler les termes importants qui ont été introduits dans la précédente phase de bibliographie, ceci afin de pouvoir envisager sereinnement la poursuite de la phase de conception du projet.

3.1 Rappels

Nous allons tout d'abord rappeler les différentes notions qui ont déjà été abordées durant l'étape de bibliographie et d'analyse des besoins.

Trois d'entre elles nous semblent être les plus importantes : les notions de drivers général, ODBC et JDBC.

3.1.1 Driver

Nous avions vu dans la partie bibliographique que le « terme driver » désigne généralement un exécutable permettant à un logiciel haut niveau de communiquer avec un périphérique matériel. Cet exécutable sert en quelque sorte de traducteur entre le logiciel et le périphérique.

Si la définition ci-dessus est correcte pour la grande majorité des drivers, elle ne s'applique pas pour le terme driver tel que nous l'entendrons dans ce rapport.

Dans le cadre de ce projet, le terme driver est utilisé pour désigner un exécutable permettant l'échange de données, non pas entre une application et un périphérique matériel, mais entre entre deux applications.

Plus précisément, l'échange de données est effectué entre un SGBD, ou logiciel de traitement de données, et un logiciel disposant d'une base de données. Le driver permet alors au SGBD d'accéder à la base de données de l'autre logiciel, en lecture ou en écriture.

Dans notre rapport bibliographique, nous avions distingué différents type de driver accomplissant cette tâche : ODBC, JDBC, OLE DB, ADO, ...

3.1.2 ODBC

ODBC est un ensemble API/pilote permettant la communication entre des applications disposant d'une bases de données et des SGBD, ou logiciels de traitement de données.

Ce type de driver développé par Microsoft a l'avantage d'être un standard parmi les drivers. Étant le plus ancien et le plus populaire des types présentés ici, la quasi-totalité des SGBD et des logiciels traitant des bases de données est compatible avec ce type de driver.

L'API ODBC est une interface procédurale : ce n'est pas un protocole à destination d'un serveur, comme peut l'être par exemple le protocole HTTP. L'API est accessible aux applications clientes sous la forme de bibliothèques à liaison dynamique.

Ces bibliothèques, conçues habituellement par le fournisseur de la base, constituent le pilote ou driver de la base en question et doivent être installées sur le poste client.

Le gestionnaire ODBC est présent sur de nombreuses plates-formes, notamment des plates-formes Windows, avec odbc32, et de type UNIX, avec iodbc pour Linux.

La technologie ODBC permet d'interfacer de façon standard une application à n'importe quel serveur de bases de données, pour peu que celui-ci possède un driver ODBC (la quasi-totalité des SGBD du marché possède un tel pilote).

3.1.3 JDBC

Très similaire au type ODBC, le type de driver JDBC est également capable de se connecter à la grande majorité des SGBD du marché.

Les drivers JDBC sont nativement codés en Java, ce qui les rend particulièrement portables.

On notera tout de même que les drivers JDBC ont le désavantage d'être moins bien pris en charge par Windows que les drivers ODBC. S'il est possible de lire une base de données Excel depuis un logiciel utilisant ODBC, il est en revenche impossible de lire une base de données extérieure depuis Excel. Ce désaventage majeur rend ce type de driver inutilisable pour la réalisation de notre projet.

Récapitulatif des rappels bibliographiques

Le rappel de ces notions de base nous permet de nous recentrer au niveau des drivers, domaine principal de notre projet.

Le rappel de l'étude de JDBC permet notamment de redéfinir les solutions possibles répondant à notre projet.

3.2 Compléments d'information

Après avoir défini les notions de base qui ont été abordées durant la phase de bibliographique, nous allons désormais introduire les nouveaux termes que nous avons pu aborder tout au long de cette phase de conception : les différents moyens de connexion entre ODBC et JDBC, ainsi que JNI.

3.2.1 Bridge JDBC-ODBC (Fig. 3.1)

En continuant nos recherches sur les drivers utilisant le protocole JDBC, développé par SUN, nous avons découvert que ceux-ci se décomposent en quatre familles distinctes disposant d'architectures différentes.

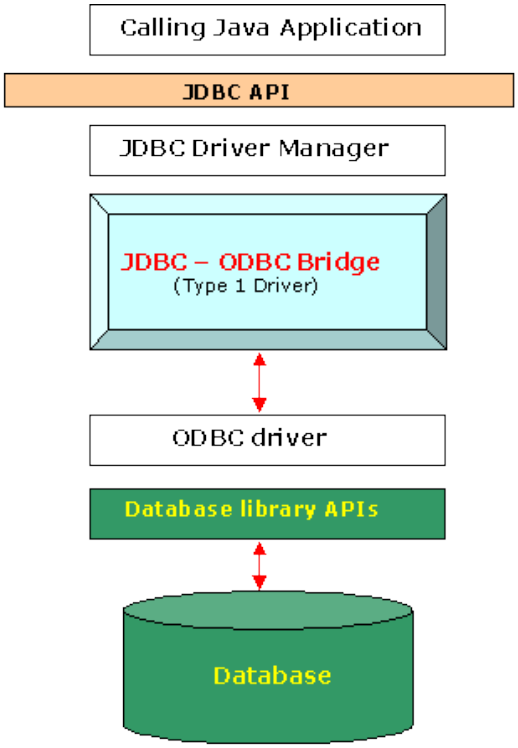


FIG. 3.1 – Schéma d'un pont JDBC-ODBC [6]

L'une d'entre elles nous a particulièrement attiré notre attention : le « bridge JDBC-ODBC ».

Ce driver accède à la base de données en passant par l'intermédiaire du protocole ODBC. Les appels formulés par le logiciel de requêtes utilisant le protocole JDBC sont alors convertis en appels ODBC avant d'être envoyés au SGBD (ici, la structure de données) dont on veut exporter les données.

De même, les données renvoyées par le SGBD doivent être traduites avec le protocole ODBC puis JDBC avant d'être exploitables.

Comme on peut l'imaginer, ce protocole est particulièrement lent. En revanche, il possède l'avantage de pouvoir exporter des données depuis tous les SGBD compatibles avec ODBC.

Comme on peut le constater dans ce bref descriptif, la communication effectuée par le bridge ODBC-JDBC ne correspond pas à nos attentes. En effet, le logiciel contenant la base de données communique via l'interface ODBC et le logiciel important les données communique via l'accès JDBC.

Or, Excel ne sait pas communiquer par le biais de JDBC et le logiciel de M. Guérin est développé en Java, alors que le protocole ODBC est natif C.

3.2.2 Gateway ODBC-JDBC (Fig. 3.2)

Au cours de nos recherches sur les « bridges JDBC-ODBC », nous avons découvert un type de driver répondant exactement à nos besoins pour le projet : le Gateway ODBC-JDBC.

Le « Gateway ODBC-JDBC » est un driver ODBC, écrit en C, et qui utilise JNI pour s'interfacer et se mettre en relation avec des appels JDBC.

Ce type de driver implémente l'interface JDBC côté le logiciel contenant la base de données, et l'interface ODBC côté le logiciel demandant l'importation de données.

Le « Gateway ODBC-JDBC » serait donc une solution parfaitement adaptée à notre projet :

- Excel communique via l'interface ODBC du driver.
- La requête ODBC est traduite en requête JDBC par le driver
- Le logiciel de M. Guérin communique via l'interface JDBC, natif Java.

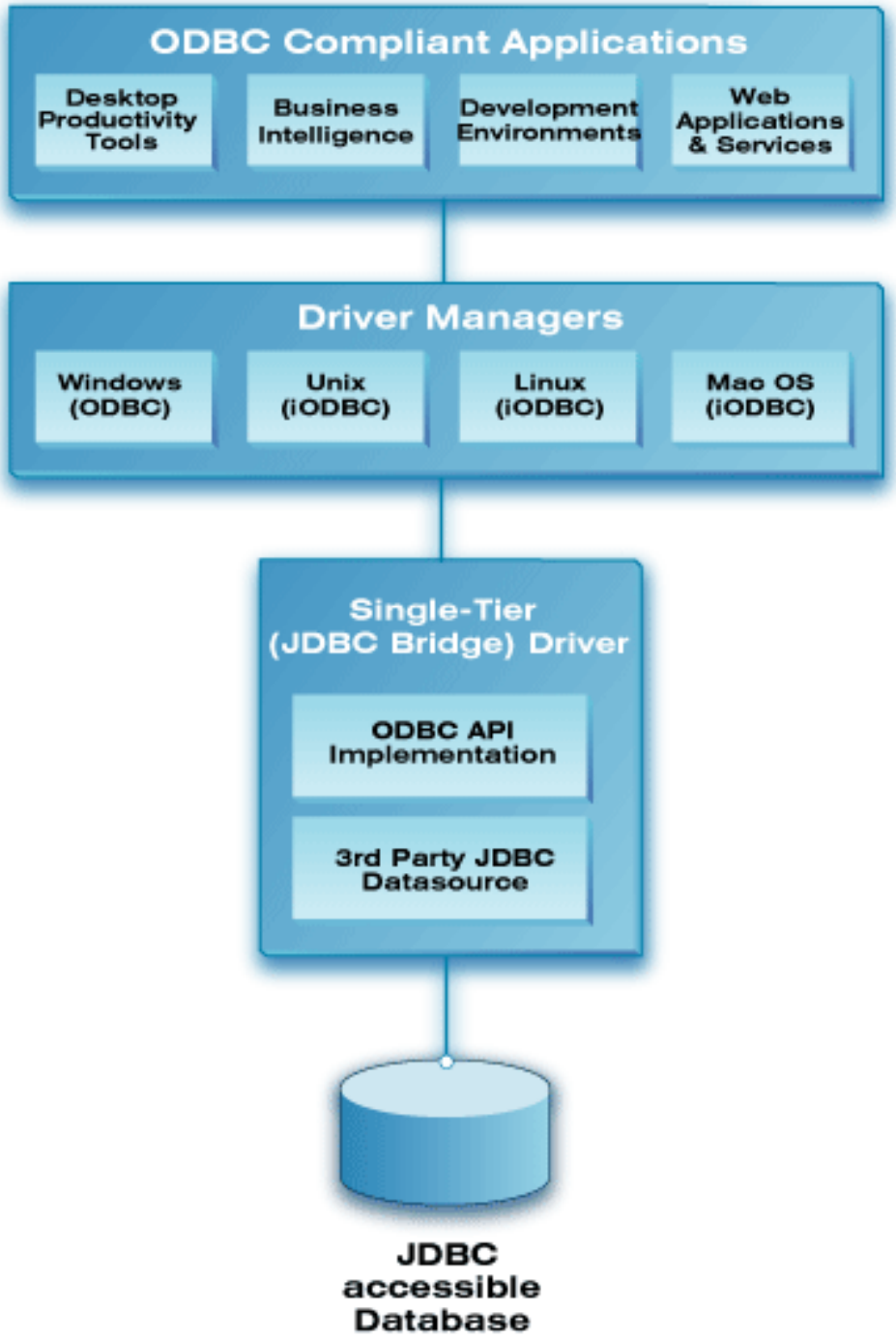


FIG. 3.2 – Schéma d'un driver ODBC-JDBC de type 1 [21]

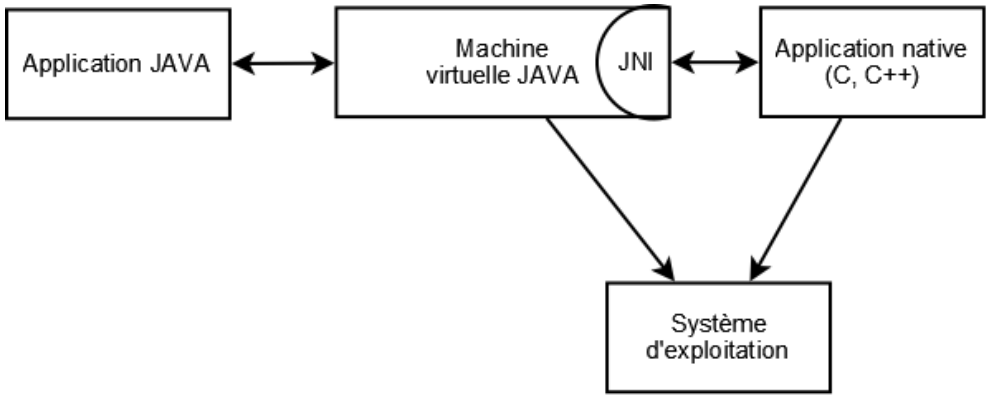


FIG. 3.3 – Schéma de communications avec JNI [22]

3.2.3 JNI (FIG. 3.3)

Le JNI (*Java Native Interface*) est un framework qui permet à du code Java s'exécutant à l'intérieur de la *Java VIrtual Machine*) d'appeler et d'être appelé par des applications natives (c'est-à-dire des programmes spécifiques au matériel et au système d'exploitation de la plate-forme concernée), ou avec des bibliothèques logicielles basées sur d'autres langages (C, C++, assembleur, ...).

Voici quelques exemples d'utilisation de la JNI :
- Certaines fonctions du matériel ou du système d'exploitation ne sont pas implémentées dans les bibliothèques Java
- Pouvoir s'interfacer avec des applications écrites dans d'autres langages
- Pour les applications temps réel, utiliser un langage compilé (c'est-à-dire du code natif) sera plus rapide que de passer par le *bytecode* de Java.

Cette technologie peut être très utile dans plusieurs cas :
- pour des raisons de performance
- utiliser des composants éprouvés déjà existants

L'inconvénient majeur de cette technologie est d'annuler la portabilité du code Java.

L'utilisation d'objets écrits en Java, à partir d'un programme écrit en C, demande de connaître la représentation en mémoire des objets créés par la machine virtuelle Java. Ensuite, au moment de l'exécution, il faut activer ces objets Java, c'est à dire lancer une machine virtuelle, y charger une classe, puis un objet, afin d'accéder à un champ ou une méthode.

Tout ce qui concerne ces commandes se trouve dans l'environnement JNI qui comprend évidemment une partie Java (commandes `javah`, `javap`) et une partie C (fichier `jni.h`, ...).

La mise en oeuvre de JNI nécessite plusieurs étapes :
- la déclaration et l'utilisation de la ou des méthodes natives dans la classe Java
- la compilation de la classe Java
- la génération du fichier d'en-tête avec l'outil `javah`
- l'écriture du code natif en utilisant entre autre les fichiers d'en-tête fourni par le JDK et celui généré précédemment
- la compilation du code natif sous la forme d'une bibliothèque

Le format de la bibliothèque est donc dépendante du système d'exploitation pour lequel elle est développée : `.dll` pour les systèmes de type Windows, `.so` pour les système de type Unix, ...

Récapitulatif du complément d'information

Toutes ces nouvelles notions auront été abordées durant la phase de conception.

Elles viennent s'ajouter aux termes déjà décrits dans le rappel de bibliographie.

Récapitulatif du complément de bibliographie

La réalisation de ce document nous a permis de mieux comprendre les caractéristiques et le fonctionnement des drivers.

Si son but était de nous aider à déterminer le type de driver que nous avions à choisir pour la suite du projet, nous nous sommes aperçus tardivement que le driver JDBC ne répondait pas à nos attentes.

En effet, si les drivers JDBC sont capables de se connecter à un très grand nombre de bases de données et d'être utilisés par des logiciels de traitement de requêtes tels qu'Open Office, nous n'avons pas réussi à utiliser ce type de driver avec le module d'importation de données d'Excel.

Après quelques recherches, nous avons compris que le driver JDBC, bien que capable de se connecter à un grand nombre de structures de données, est surtout utilisé dans le dévelopement de logiciels nécessitant d'importer des données. Il semble difficile, sinon impossible, de l'utiliser pour importer des données vers un grand nombre de logiciels déjà existants tels qu'Excel et Access.

Tout ce complément d'information nous sera très utile, car il sera réutilisé lors de la recherche de solutions.

Chapitre 4

Recherche de solution

Après avoir rappelé et introduit les notions bibliographiques inhérentes au projet, nous nous lançons désormais à une recherche approfondie de solutions afin de pouvoir poursuivre notre projet.

4.1 ODBC-JDBC

L'une des solutions que nous avons étudiées est l'utilisation d'un type spécial de driver ODBC, permettant de connecter ODBC à JDBC : le « Gateway ODBC - JDBC ».

L'interêt d'utiliser ce type de driver était double :

- Pouvoir nous connecter à une source de données compatible avec JDBC
- Pouvoir connecter le driver en utilisant le connecteur ODBC présent dans les modules d'importation des divers logiciels de gestion de données du commerce.

4.1.1 Objectif

L'objectif de notre recherche était de trouver un projet de « Gateway ODBC - JDBC », *Open Source* utilisable dans un projet commercial.

Nos recherches se sont évidemment faites par l'intermédiaire du Net. Celles-ci se sont révélées particulièrement infructueuse. Nous avons cependant pu mettre la main sur quelques drivers :

- celui de *EasySoft* [23]
- une source sur SourceForge [24]

- le driver créé par *Firebird* [25] qui permet, à l'origine, de se connecter au SGBD propriétaire de *Firebird*
- les drivers commerciaux de *EasySoft* [23], *OpenAccess* [26] et *OpenLink* [21]

4.1.2 Résultat

Il nous a été particulièrement difficile de trouver des informations et des projets sur ce type de driver. De plus, aucun des drivers cités précédement n'a pu être exploité.

En effet, soit le driver est payant (*EasySoft*,*OpenAccess*,*OpenLink*), soit le projet de développement du driver est mort (« dead project » sur SourceForge), ou bien inutilisable car trop complexe et trop peu commenté (driver de *Firebird*).

Récapitulatif concernant ODBC-JDBC

Après avoir constaté qu'aucun des drivers permettant la connexion entre ODBC et JDBC n'était utilisable, nous avons abandonné cette piste pour étudier d'autres possiblités.

4.2 ODBC

L'autre solution que nous avons étudiée est l'utilisation d'un driver ODBC, mais en étudiant en priorité une source déjà disponible sur Internet.

4.2.1 Objectif

Notre objectif en étudiant puis en utilisant ce type de driver était de pouvoir comprendre le fonctionnement d'un driver ODBC, pour ensuite réutiliser tout ou une partie du code trouvé.

Nos recherches se sont également faites par l'intermédiaire du Net. Nous y avons trouvé plusieurs drivers qui correspondaient à nos attentes :

- le connecteur ODBC/MySQL de *iODBC* [27]
- le driver MySQL original [28]

Nous avons également étudié le SDK ODBC [29] et le site MSDN [30] de Microsoft.

Pour réaliser la communication entre un driver ODBC et une application Java, nous sommes reinseignés sur les protocoles de communication entre des applications C et Java

4.2.2 Résultat

L'étude des drivers et des aides fournies par Microsoft nous ont permis de mieux comprendre le fonctionnement d'un driver ODBC.

Pour réaliser la communication entre le driver C et un logiciel Java, nous avons pu choisir entre deux solutions :

- Effectuer la communication par le biai de DDE (Dynamic Data Exchange), une interface de communication entre programme C, et utiliser JNI pour implanter DDE coté application Java
- Utiliser un protocole réseau de type Socket pour établir la communication entre les deux applications

Récapitulatif concernant ODBC

L'étude du SDK de Microsoft et du site MSDN nous semble être la piste à privilégier pour pousuivre le projet. En revanche, les codes sources des drivers ODBC trouvés sur Internet sont bien trop compliqués et trop peu documentés pour servir de base à notre driver.

Pour ce qui est du protocole de communication, nous utiliserons le protocole Socket, plus simple et plus propre que la méthode utilisant DDE.

Récapitulatif de la recherche de solution

La recherche de solutions pour notre projet nous a permis d'éliminer plusieurs pistes que nous pensions pourtant être justes.

La solution nous semblant la plus pertinente est l'utilisation d'un driver ODBC. C'est ainsi que nous sommes passés à la recherche d'un driver de type ODBC, codé en langage C.

Chapitre 5

Dossier de conception

Ce chapitre explicite la conception du pilote qu'il nous est demandé de réaliser ainsi que des différents composants nécessaires à son fonctionnement.

Le projet sera abordé par une vue d'ensemble puis découpé en différentes étapes de conception. Chaque étape sera traité plus en détail dans les différentes parties de ce dossier.

Enfin, nous estimerons le temps à passer sur chaque étape et exposerons comment sera organisé notre temps de travail.

5.1 Vue générale de l'architecture du projet (Fig. 5.1)

Avant de rentrer dans les détails de la conception, nous commencerons par prendre le projet dans son ensemble.

Cette première analyse du problème permet de décrire d'une part les éléments existants sur lesquels nous nous appuierons, et de définir d'autre part les éléments que nous devrons créer ainsi que leurs fonctionnalités.

5.1.1 Contexte du projet

Pour réaliser la conception de ce projet, nous nous appuierons sur différents éléments existants :

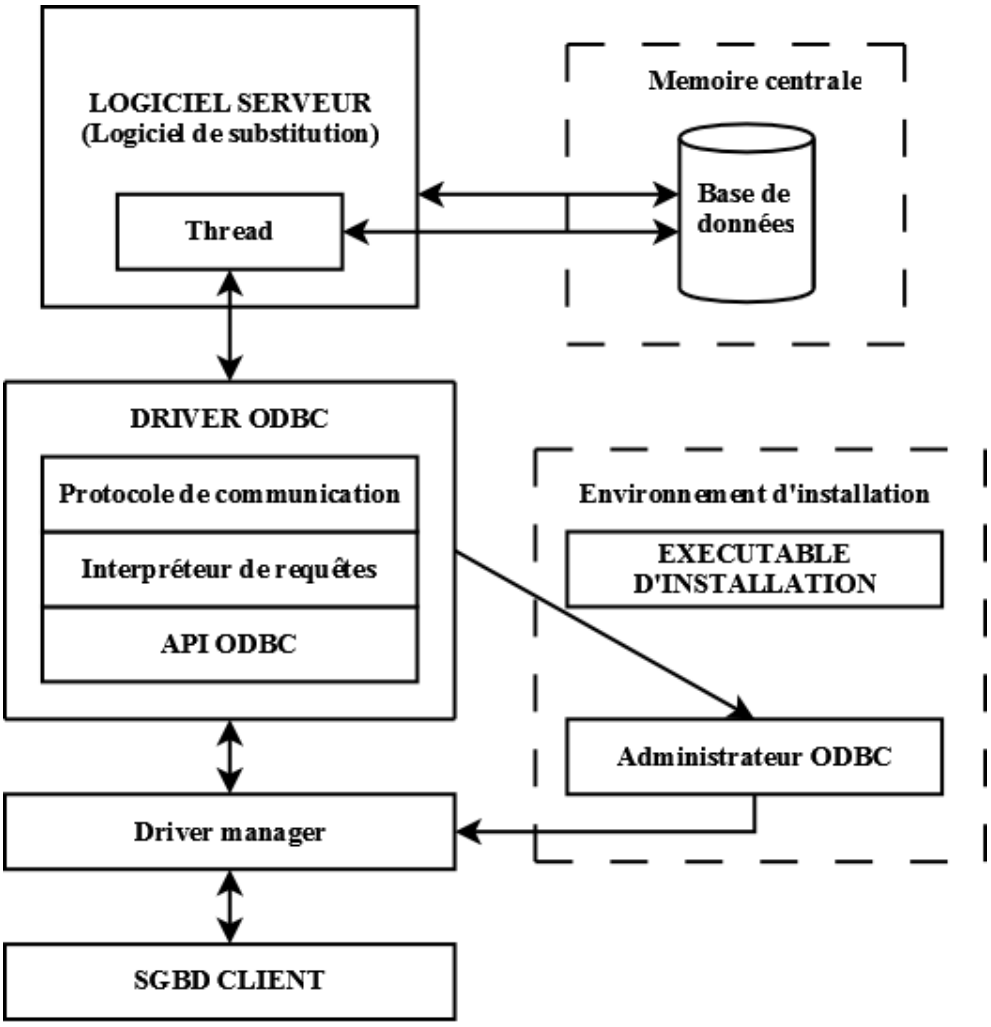


FIG. 5.1 – Schéma général, représentant l'architecture du projet

Le SGBD client

C'est l'application dans laquelle on veut importer des données. Elle doit implémenter un module d'importation utilisant l'API ODBC.

L'administrateur de sources de données ODBC

Il permet de visualiser tous les pilotes ODBC connus par le système d'exploitation.

Pour chaque pilote, il offre la possibilité de configurer la position de la source de données sur laquelle on souhaite se connecter, ainsi que différents paramètres.

Le gestionnaire de pilote ODBC (*driver manager*)

Lorsque des informations doivent être échangées entre un pilote et un SGBD client, celles-ci transitent par le *driver manager*.

Ce dernier pré-traite les informations et en vérifie la validité avant de les relayer à leur destinataire.

Le *driver manager* a aussi pour rôle de charger le bon pilote lors d'une demande de connexion.

Le logiciel serveur (logiciel de Mr Guérin)

Il dispose d'une base de données en mémoire centrale à laquelle nous souhaitons avoir accès.

Nous devrons par la suite ajouter des fonctionnalités à ce logiciel pour le rendre accessible au pilote.

5.1.2 Modules à implémenter (Fig. 5.2)

Le pilote ODBC

Le centre de notre conception réside en l'implémentation du pilote ODBC.

On peut découper le pilote en trois parties distinctes :

- L'interface ODBC : elle consiste en l'implémentation des fonctions de l'API ODBC.

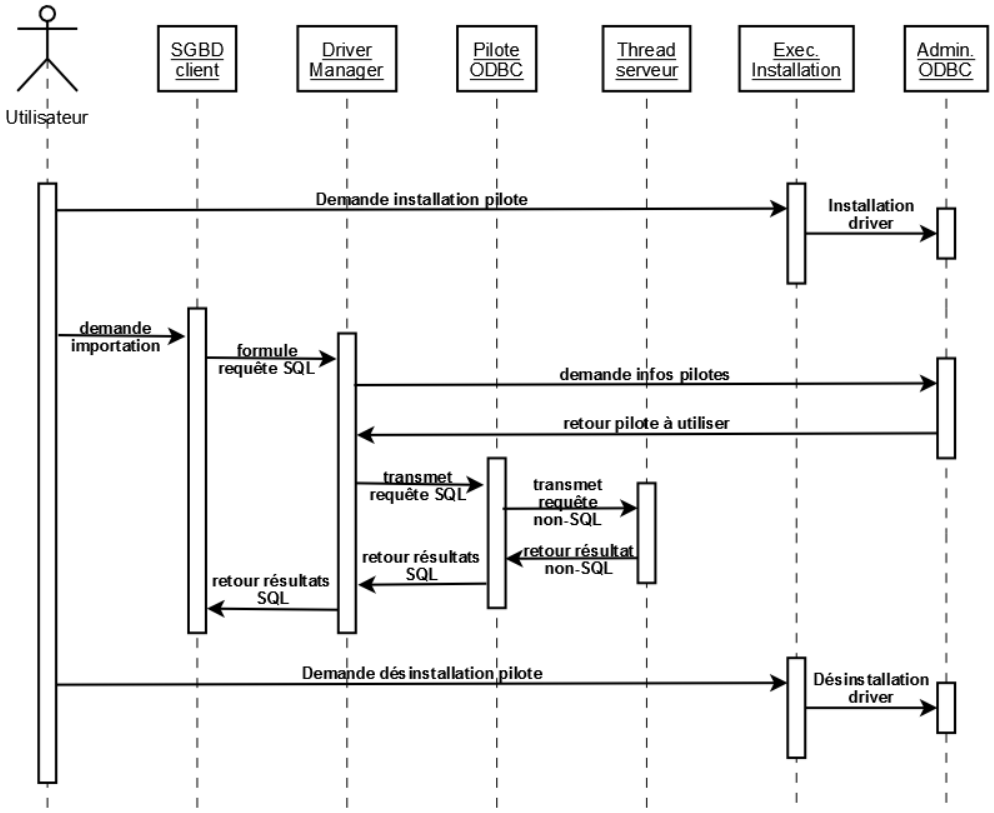


FIG. 5.2 – Diagramme UML de séquences

Elle permet d'établir la communication entre notre pilote et le *driver manager*. Elle est également nécessaire pour que le pilote puisse être reconnu par le système d'exploitation.

- L'interpréteur de requêtes : il doit pouvoir traiter les requêtes SQL transmises par le SGBD client et les transformer en requêtes non-SQL facile et rapide à analyser.
  N'ayant pu trouver d'interpréteur SQL, il nous faudra réaliser notre propre interpréteur de requêtes. Celui-ci devra au minimum être capable de gérer des requêtes simples de la forme « `SELECT * FROM 'Table'` »
- Un protocole de communication avec le *thread* du logiciel serveur.
  Ce protocole devra être capable de transmettre au *thread* les requêtes non-SQL à executer et d'en récupérer les résultats.

Le logiciel de substitution

Le logiciel de M. Guérin étant encore en cours de développement, nous devrons réaliser un logiciel de substitution disposant d'une base de données à laquelle nous devrons accéder par le biais du pilote ODBC.

Le *thread* serveur

Afin de ne pas paralyser le logiciel serveur, un *thread* en attente d'appels de la part du pilote devra être implémenté.

Le *thread* a deux fonctionnalités principales :

- Il devra être capable de communiquer avec le pilote
- Il devra traiter des requêtes non-SQL et de gérer les conflits d'accès à la base de données

Le module d'installation

Il permet d'automatiser l'installation du pilote ODBC dans le système d'exploitation.

Récapitulatif de la vue générale

Cette première analyse du problème nous a permis de décrire les éléments existants sur lesquels nous nous appuierons, et ceux que nous devrons créer ainsi que leurs fonctionnalités.

Elle pose donc les premières bases concrètes du projet.

5.2 Conception des différentes parties

Cette partie a pour but de décrire plus en détail chaque élément que nous devrons concevoir.

A chaque fois, nous verrons :
- quels sont les outils qui nous permettrons de réaliser l'élément
- quel est le rôle de l'élément
- quelle méthode nous comptons mettre en œuvre pour le concevoir
- comment nous validerons son bon fonctionnement

5.2.1 Le pilote ODBC (Fig. 5.3)

Comme décrit précédemment, le pilote ODBC se divise en 3 parties :
- L'interface ODBC coté client
- L'interpréteur de requêtes
- Un protocole de communication vers le serveur

Pilote « squelette » ODBC

Outils et bibliothèques utilisés :

- Bibliothèque de l'API ODBC : `sqlext.h`
- Exemple de pilote GODBC [31]
- Outil de test de pilote ODBC du Platform SDK de Microsoft
- Documentation sur ODBC du MSDN2

Fonctionnalités du module :

Avant toute étape, nous commencerons par implémenter une version « squelette » du pilote ODBC.

Cette version aura pour but de nous assurer que notre pilote est capable d'établir une communication entre un SGBD client et notre pilote. Cela nous permettra également de regarder plus en détail les structures que doivent prendre les requêtes et les résultats au niveau de cette API.

Méthode de conception :

La conception de ce pilote « squelette » passera par l'implémentation des différentes fonctions de l'interface ODBC (Cf. Annexe I).

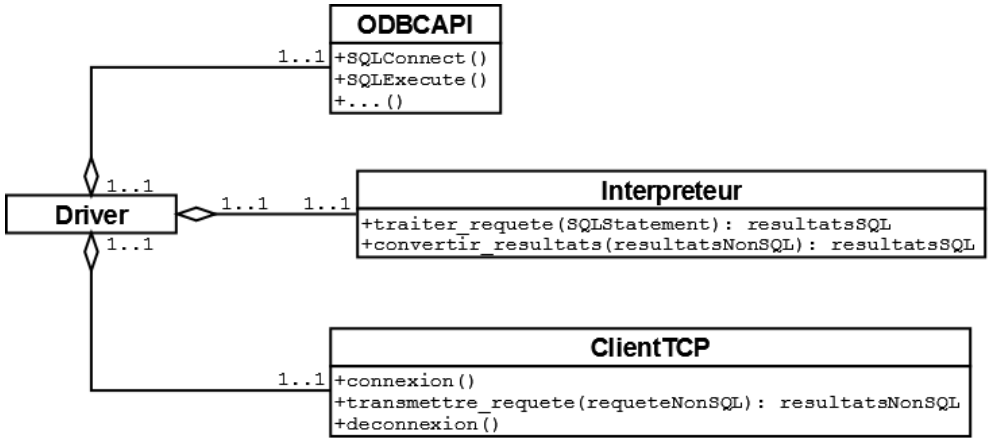


FIG. 5.3 – Diagramme UML de classes, pour le pilote ODBC

Dans le corps de chaque fonction, nous implémenterons quelques lignes de débogue. Celles-ci afficheront des boîtes de dialogue indiquant quelle fonction s'est exécutée et quelles étaient les valeurs des différents attributs de la fonction.

Les corps des fonctions implémentés dans cette interface seront complétés au fur et à mesure de notre avancée dans la phase de conception.

Tests de validité :

Pour tester la validité de notre pilote squelette, nous le compilerons sous forme de *dll* et l'installerons à la main dans la base de registre et différents fichiers du système d'exploitation.

Nous utiliserons ensuite un outil de jeu d'essais pour pilote ODBC fourni dans le « Platform SDK » de Microsoft afin d'effectuer des tests unitaires.

Enfin, nous vérifierons s'il est bien possible d'appeler ce pilote depuis Excel.

Interpréteur de requêtes

Outils et bibliothèques utilisées :

– Bibliothèque de l'API ODBC : `sqlext.h`

Fonctionnalités du module :

Une étape cruciale pour la conception du pilote est l'implémentation d'un interpréteur de requêtes.

Comme dit dans la partie I), l'interpréteur de requête que nous implémenterons sera un interpréteur SQL très simple, capable au moins d'analyser des requêtes du type « `SELECT * FROM 'Table'` ». Nous pensons que se limiter à des requêtes de ce type sera suffisant pour répondre aux besoins qui nous ont été formulés.

L'interpréteur aura deux fonctionnalités principales : traduire les requêtes SQL en requêtes non-SQL facilement interprétables par le logiciel serveur, et modifier la structure des résultats non-SQL en résultats SQL utilisable par l'API ODBC.

Méthode de conception :

L'interpréteur de requête SQL doit donc disposer d'une première fonction de traitement de requêtes. Cette fonction prend en paramètre une requête SQL arrivée par l'API ODBC sous forme de « SQL statement ». Elle doit déterminer si cette requête est valide, puis la convertir en requête non-SQL, sous forme d'une liste de paramètres qui sera envoyée à l'application serveur.
Une seconde fonction, prenant en paramètre les résultats de l'application serveur après exécution d'une requête, aura pour rôle de convertir les résultats non-SQL du serveur en résultats SQL conformes à l'API ODBC.

Selon les besoins des fonctions de l'API ODBC, l'interpréteur SQL devra également implémenter différentes fonctions permettant de donner des informations sur un résultat SQL : la taille, le nombre de colonnes, le nom des attributs, la clé primaire, ...
Dans un premier temps, nous nous contenterons de convertir les requêtes SQL arrivées au niveau de l'API ODBC en requêtes non-SQL et de renvoyer des jeux de résultats prédéfinis. L'interpréteur SQL sera intégré au pilote, et les fonctions de l'API correspondantes seront complétées.
Nous reviendrons sur l'interpréteur SQL pour achever la fonction de renvoi de données une fois que les modules de communication et de traitement de résultats côté application serveur seront réalisés.

Tests de validité :

Nous vérifierons que les requêtes transmises par Excel sont bien transmises par le pilote et correctement interprétées par cet interpréteur.

Nous nous assurerons également que Excel peut récupérer les jeux de résultats transmis depuis l'interpréteur.

Protocole de communication avec le *thread* serveur

Outils et bibliothèques utilisées :

- Bibliothèque de Socket : `inet.h` ou `winsock.h`
- Classes de Socket en java : InetAdress et Socket dans `java.net`

Fonctionnalités du module :

Ce protocole de communication devra être capable de connecter le pilote au *thread* de l'application serveur. Cette communication permettra d'échanger les requêtes non-SQL et les résultats de retour.

Pour réaliser cette communication, nous implémenterons un protocole basé sur les Socket réseaux.

Cette solution a l'avantage d'établir de manière simple et fiable la communication entre notre pilote natif C++ et le logiciel serveur développé en Java.

En reprenant l'architecture Socket, le *thread* du logiciel serveur sera alors un serveur TCP, continuellement en attente de connexion.

Le pilote se comportera comme un client. Enfin, les requêtes et les résultats seront transmis sous forme de flux par ce protocole.

Méthode de conception :

Pour développer cette partie du pilote, nous développerons à la fois une application serveur simple en Java et la partie cliente du pilote en C++. L'application serveur sera réutilisée lors de l'implémentation du *thread* de l'application serveur.

Nous implémenterons les différentes fonctions nécessaires à la communication par Socket en TCP : attente de connexion, connexion, envoie, réception, déconnexion, ... Une fois développés, les clients et serveurs seront intégrés respectivement au pilote et au *thread*.

Tests de validité :

Pour valider cette partie, nous nous assurerons que le client est bien capable de se connecter, se déconnecter et de communiquer avec le serveur, et que le serveur est capable de répondre.

5.2.2 Réalisation d'un logiciel de substitution

Outils et bibliothèques utilisées :

- Classe de lecture de fichiers : `FileReader` dans `java.io`

Fonctionnalités du module :

Le logiciel de M. Guérin étant encore en développement, nous réaliserons un logiciel de substitution, développé en Java, disposant d'une base de donnée en mémoire centrale.

Ce logiciel nous servira pour :

- concevoir le *thread* client permettant de recevoir les requêtes non-SQL
- tester et valider notre pilote et notre *thread*
- rédiger un document expliquant comment intégrer de notre travail dans un projet

Méthode de conception :

Ce logiciel chargera en mémoire centrale des données, provenant d'un fichier CSV, organisées sous forme de listes de structures avec :

- Une liste correspondra à une table
- Le 1er élément de la liste correspondra au nom des attributs
- Tout autre élément de la liste correspondra à un t-uple de la table

5.2.3 Le *thread* serveur (Fig. 5.4)

Outils et bibliothèques utilisées :

- Classes de Socket en java : InetAdress et Socket dans `java.net`
- Classe gérant les *threads* en java : *thread* du package `java.lang`

Fonctionnalités du module :

Pour accéder à la base de données, le pilote ODBC doit faire appel au logiciel serveur.

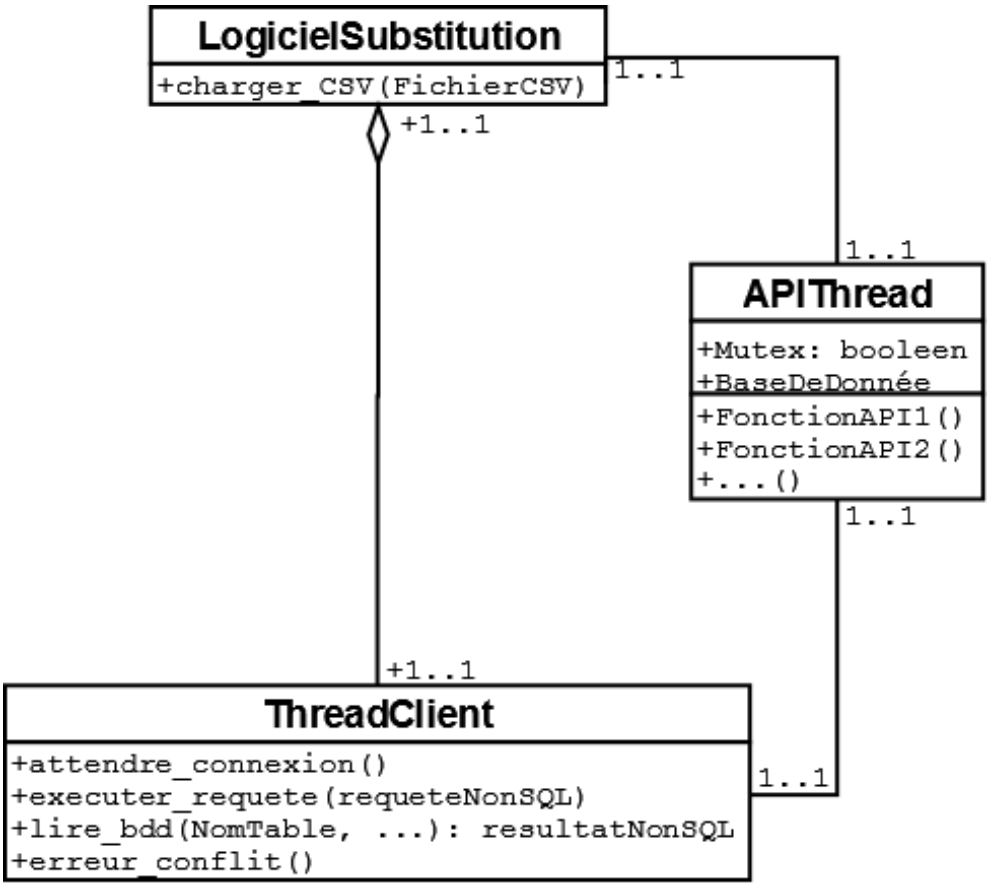


FIG. 5.4 – Diagramme UML de classes, pour le *thread* serveur

Comme vu précédemment, la communication entre le pilote et le logiciel serveur sera effectuée par le biais d'un protocole de type Socket TCP. L'inconvénient majeur de cette solution est que, pour espérer établir une connexion, le logiciel serveur doit rester en état d'attente de connexion.
Pour palier à ce problème d'état paralysant, la solution adoptée est de lancer un *thread* depuis l'application serveur. Ce *thread* sera chargé de rester en position d'attente de connexion et de recevoir la connexion du pilote. Ainsi, le reste de l'application serveur pourra continuer à fonctionner normalement.
Le *thread* de l'application serveur ne sera pas seulement utilisé pour communiquer avec le pilote. Il devra également exécuter les requêtes non-SQL qui lui seront communiquées pour en retourner les résultats.

Méthode de conception :

Le *thread* devra être lancé depuis le logiciel de substitution. Afin de rendre le *thread* rapide à implanter dans un projet, une classe contiendra les différentes variables partagées par le *thread* et le logiciel serveur, ainsi que différentes fonctions que le logiciel serveur devra utiliser pour faire fonctionner le *thread*.

Tel que vu précédemment, la partie serveur TCP du *thread* sera réalisée en même temps que le client TCP du pilote ODBC. Nous intégrerons cette première fonctionnalité au *thread*.

Afin d'exécuter les requêtes qui lui seront transmises, le *thread* devra connaître la position en mémoire centrale des différents éléments de la base de données du logiciel serveur. Ayant connaissance de la base de données, le *thread* devra implémenter une fonction lui permettant de récupérer les données d'une table.
Cette fonction prendra en paramètre les différents arguments d'une requête non-SQL envoyée par le serveur, et prendra pour valeur de retour le résultat de la requête sous forme de flux d'information.

Pour éviter les problèmes de lectures sales, il faudra s'assurer que la base de données n'est pas en train d'être modifiée par l'application serveur lorsque le *thread* veut la lire. Aussi, si l'application serveur a besoin d'écrire sur la table lue par le *thread*, alors cette lecture doit être abandonnée et se résoudre par un échec (nous considérons ici que l'application a la priorité sur le *thread*).

Pour régler ces problèmes, le *thread* et l'application serveur partageront une ou plusieurs variables d'exclusion mutuelle. Si possible, nous appliquerons le principe d'exclusion mutuelle par table de la base de données.

Tests de validité :

Pour tester le *thread*, une base de données contenant des tables de test sera mise en mémoire centrale par le logiciel de substitution. Nous enverrons alors un jeu de requêtes au *thread* et vérifierons si celui-ci renvoie bien les résultats attendus.

5.2.4 Exécutable d'installation

Outils et bibliothèques utilisées :

- Bibliothèque d'installation de pilote ODBC : `odbcinst.h`
- Bibliothèque Windows pour utiliser des fenêtres win32 : `windows.h`

Fonctionnalités du module :

La dernière partie de conception de ce projet sera la réalisation d'un exécutable d'installation du pilote. Il devra permettre d'automatiser l'installation et la désinstallation du dll dans le système d'exploitation.

Pour configurer les différents paramètres du pilote, une IHM guidée sera mise en place.

Tests de validité :

L'installateur devra être capable d'installer, de configurer et de désinstaller notre pilote ODBC.

5.2.5 Tests d'intégrité et validation du projet

Nous réaliserons une dernière série de tests finaux pour s'assurer que l'ensemble du projet fonctionne correctement :

- Le pilote ODBC sera installé par l'exécutable d'installation sur différentes machines.

– Nous l'utiliserons pour transférer des tables de base de données, générées par le logiciel de substitution depuis des fichiers CSV, vers différents SGBD. Nous vérifierons alors l'intégrité des données véhiculées.
– Le pilote sera désinstallé par l'exécutable d'installation.

Une fois ce test passé avec succès, nous pourrons considérer notre produit comme valide.

Récapitulatif de la conception des parties

Après avoir défini dans les grandes lignes les éléments que nous devrons créer ainsi que leurs fonctionnalités (ainsi que ceux existants sur lesquels nous nous appuierons), nous venons désormais décrire plus en détail chacun des éléments à réaliser.

5.3 Organisation du temps de travail

Afin de pouvoir estimer le temps à consacrer sur chacune des tâches, nous nous appuierons sur un planning prévisionnel.

Ce genre de planning montre comment organiser notre temps, l'ordre dans lequel effectuer les tâches et la date d'échéance.

5.3.1 1er élément

Elément :

Pilote « squelette » ODBC

Eléments prérequis :

Pas d'élément prérequis

Difficulté :

La difficulté de cette partie réside majoritairement dans la compréhension du rôle de chaque fonction de l'API ODBC, de ses entrées, et de ses valeurs de retour

Estimation du temps d'implémentation (en heures) :

(difficulté *5), soit 25h00

5.3.2 2e élément

Elément :

Interpréteur de requêtes

Eléments prérequis :

Pilote « squelette » ODBC

Difficulté :

Cette partie a deux difficultés majeures :

- Comprendre comment une requête SQL est enregistrée dans une structure SQLStatement
- Savoir quelles fonctions de l'API sont à modifier pour permettre de transférer les requêtes SQL vers l'interpréteur SQL

Estimation :

(difficulté *4), soit 20h00

5.3.3 3e élément

Elément :

Protocole de communication avec le *thread* serveur

Eléments prérequis :

Pas d'élément prérequis

Difficulté :

Comprendre le fonctionnement des socket en C et en Java

Estimation du temps d'implémentation (en heures) :

(difficulté *2), soit 10h00

5.3.4 4e élément

Elément :

Logiciel de substitution

Eléments prérequis :

Pas d'élément prérequis

Difficulté :

Aucune

Estimation du temps d'implémentation (en heures) :

(difficulté *1), soit 5h00

5.3.5 5e élément

Elément :

Thread serveur

Eléments prérequis :

Logiciel de substitution, protocole de communication

Difficulté :

Cette partie est particulièrement compliquée de par le fait qu'elle s'appuie sur une technologie qui ne nous est pas familière : la programmation de *thread*.

Voici les difficultés que nous rencontrerons :

- Se familiariser avec la programmation de *thread* (création de *thread*, partage de variables, etc.)
- Implémenter la fonction d'exécution de requêtes et le système d'exclusion mutuelle
- Effectuer une API simple pour rendre le travail facilement implémentation dans un projet

Estimation du temps d'implémentation (en heures) :

(difficulté *5), soit 25h00

5.3.6 6e élément

Elément :

Exécutable d'installation

Eléments prérequis :

Pilote « squelette » ODBC

Difficulté :

Comprendre les fonctions de la bibliothèque d'installation ODBC, et réaliser une interface en win32

Estimation du temps d'implémentation (en heures) :

(difficulté *3), soit 15h00

Récapitulatif de l'organisation du travail (Fig. 5.5)

Le planning prévisionnel que nous venons de créer ne s'appuie sur aucune base formelle (COCOMO par exemple), mais est approximé.

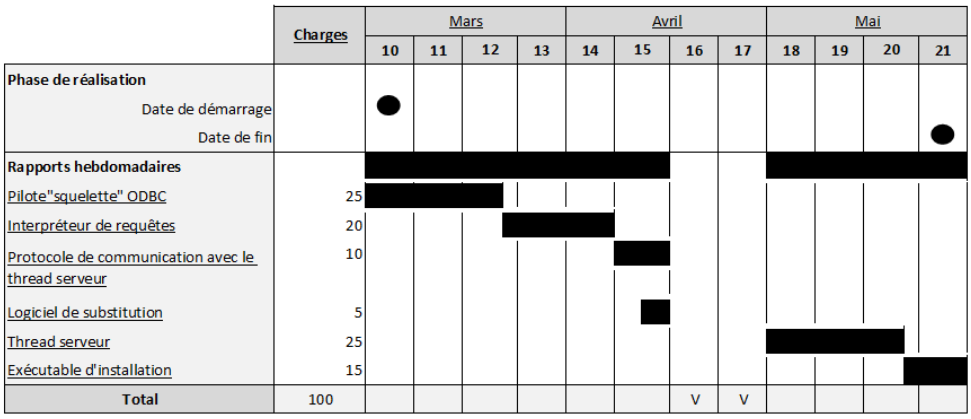


FIG. 5.5 – Planning prévisionnel de la phase de réalisation

Il nous permettra d'estimer avec une relative précision les dates auxquelles nous devrons nous conformer lors de la réalisation du projet.

Récapitulatif du dossier de conception

Ce chapitre de conception nous a permis de mieux comprendre les différentes parties logicielles et/ou matérielles entrant en jeu dans le projet.

Récapitulatif

La description de la modélisation du projet nous permet de rendre plus « concret » et de joindre l'ensemble des informations recueillies et des choix réalisés depuis la fin de la phase de bibliographie.

Cette phase de conception nous a permis de définir le logiciel, ceci nous permettant de nous préparer à l'implémentation de celui-ci, durant la prochaine étape.

Troisième partie

Réalisation

Après avoir réalisé une approche bibliographique et une analyse des besoins de notre projet, ainsi qu'après avoir étudié la conception et la modélisation de notre projet, nous allons maintenant exposer l'étape de réalisation de celui-ci.

L'objectif est d'implémenter et d'effectuer la mise en place du code informatique en s'appuyant sur la modélisation décidée à l'étape précédente.

Nous rappellerons tout d'abord les bases du projet, en redéfinissant les objectifs et les besoins le concernant.

Ensuite, nous exposerons le travail effectué durant cette phase de réalisation, et tenterons de décrire précisément les erreurs auxquelles nous fûmes confrontés.

Enfin, nous présenterons les solutions possibles et à envisager afin de répondre à la finalité du projet.

Une conclusion achèvera cette phase, et servira également de conclusion générale au projet.

Nous avons ici choisi de ne pas réaliser de calendrier de travail effectivement réalisé tout au long du projet, étant donné que celui ne comporterait que trop peu d'informations réellement susceptibles de nous intéresser.

De même, réaliser des tests à ce stade du projet ne nous serait d'aucune utilité, car ils servent avant tout à vérifier le logiciel.

Chapitre 6

Notre projet et ses objectifs

Cette première partie va nous permettre de reprendre les grandes étapes d'avancement de notre projet transversal.

Nous exposerons également les résultats que nous souhaitons atteindre avec ce projet.

6.1 Rappel du travail précédemment effectué

Le travail réalisé jusqu'à maintenant dans le cadre de notre projet s'est fait en deux étapes : l'étude bibliographique et la phase de conception. Nous allons reprendre les points importants de ces deux parties.

6.1.1 Étape « Bibliographie & Analyse des besoins »

Durant cette première phase, nous avons pu étudier les plus importantes notions inhérentes à notre projet.

Nous avons plus particulièrement fait l'étude des différentes *API* existantes et réalisant l'échange de données. Parmi ces *API*, nous nous sommes notamment penchés la technologie ODBC.

Nous nous sommes également intéressés à tout ce qui entourait le résultat technique en lui-même, mais qui reste tout aussi important : la gestion des licences de distribution, le dépôt du projet et la gestion des différentes versions.

6.1.2 Étape « Modélisation »

La phase de conception a eu pour but de définir le schéma général de développement à adopter lors de la phase d'implémentation.

La création des différents diagrammes nous a permis de mieux décrire les méthodes générales de développement, et de mieux visualiser les grandes étapes à suivre pour la suite du projet.

6.2 Travail prévu pour la suite du projet dans le cadre de l'étape de réalisation

A la fin de la phase de conception, notre planning définissait notre travail attendu pour la période suivante. Ce travail était donc de développer pilote ODBC afin de le rendre fonctionnel et utilisable dans le cadre de notre projet.

À partir de ce driver squelette, plusieurs étapes se présentent alors à nous :
- reprendre le squelette du pilote
- le réutiliser en s'appuyant sur les éléments que l'on a trouvés à côté : *MSDN*, autres pilotes ODBC existants
- utiliser l'un des logiciels connus (*Excel*, par exemple) pour le tester
- le compléter à l'aide des pilotes ODBC existants (*myODBC*, le pilote de *PostGre*)

6.3 Point critique du processus d'implémentation

L'étude de ce squelette de pilote ODBC nous a permis de comprendre la façon dont il a été créé.

Nous avons pu remarquer qu'aucune fonction n'est implémentée, si ce n'est la fonction de connexion `SQLDriverConnect`.

Ainsi, nous comprenons que le squelette de pilote fourni correspond au pilote ODBC minimal. Nous aurions donc pu très bien le reconstruire nous-même, mais cela n'aurait eu aucun véritable intérêt pour le comprendre.

Arrivés à l'entame du développement, nous avons bloqué sur une étape vitale au développement du projet. En effet, ce point critique était de faire fonctionner le squelette du pilote ODBC. Nous allons voir ce qu'il en est.

Chapitre 7

Résultats obtenus

Afin de pouvoir utiliser, pour une modification future, le pilote que nous avons trouvé, nous avons suivi et étudié les méthodes de compilations données sur le site Web que nous avons utilisé.

7.1 Utilisation du programme compilé fourni

L'auteur du driver ODBC que nous étudions, fournit plusieurs outils qui permettent d'utiliser ce pilote.

Ces outils que nous utilisons sont :

- un pilote ODBC pré-compilé, sous format DLL (bibliothèque de fonctions)
- un couple de programmes exécutables permettant l'installation et la désinstallation du pilote
- une méthode décrivant l'installation manuelle du pilote sous *Windows*

Nous choisissons donc d'utiliser ces différents outils qui nous sont fournis par l'auteur du driver.

7.1.1 Installation automatique

Nous commençons la mise en place du pilote pré-compilé en utilisant l'exécutable d'installation fourni.

Celui-ci demande le nom qui sera utilisé lors de sa création.

1. L'utilisateur doit disposer des fichiers suivants :
   - la bibliothèque de fonctions, au format `DLL`
   - l'exécutable `ODBCreg`, qui permet de gérer manuellement l'installation et la désinstallation du pilote ODBC
   - les deux fichiers de commande (*batches*) `ODBCregDev` et `ODBCuregDev`
2. L'utilisateur doit ensuite installer le pilote ODBC, en utilisant le *batche* `ODBCregDev`, dont il doit confirmer l'installation.
3. Afin de se servir du pilote ODBC, l'utilisateur doit se servir de son logiciel de requêtage habituel (tel qu'*Excel* ou *Access*), d'où il doit importer une source de données ODBC.
4. Lorsque l'utilisateur choisit de désinstaller le pilote ODBC, il doit utiliser le *batche* `ODBCuregDev`, dont il doit confirmer la désinstallation.

Nous remarquons que l'administrateur a bien reconnu notre pilote ODBC :

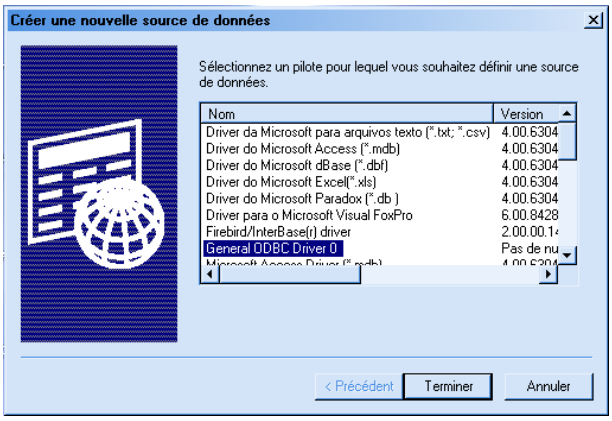
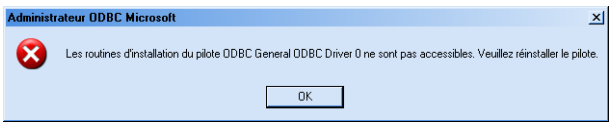

Créer une nouvelle source de données
Sélectionnez un pilote pour lequel vous souhaitez définir une source de données.
Nom
Version
Driver da Microsoft para arquivos texto (*.txt; *.csv) 4.00.6304
Driver do Microsoft Access (*.mdb) 4.00.6304
Driver do Microsoft dBase (*.dbf) 4.00.6304
Driver do Microsoft Excel(*.xls) 4.00.6304
Driver do Microsoft Paradox (*.db) 4.00.6304
Driver para o Microsoft Visual FoxPro 6.00.8428
Firebird/InterBase(r) driver
General ODBC Driver 0
< Précédent
Terminer
Annuler

FIG. 7.1 – Driver reconnu

Mais lors de la vérification de son installation, nous remarquons qu'il a été impossible pour le système d'exploitation de créer le DNS.


Administrateur ODBC Microsoft
Les routines d'installation du pilote ODBC General ODBC Driver 0 ne sont pas accessibles. Veuillez réinstaller le pilote.
OK

Nous en concluons donc que l'exécutable fourni ne permet pas d'installer le pilote fourni.

7.1.2 Installation manuelle

L'autre solution fournie par l'auteur, est de mettre en place soi-même le pilote pré-compilé sous *Windows*.

Pour cela, nous devons modifier « à la main » le registre du système d'exploitation, représenté par le programme `regedit`.

Nous suivons donc les instructions données, qui consiste à inscrire les informations du pilote au cœur du système.

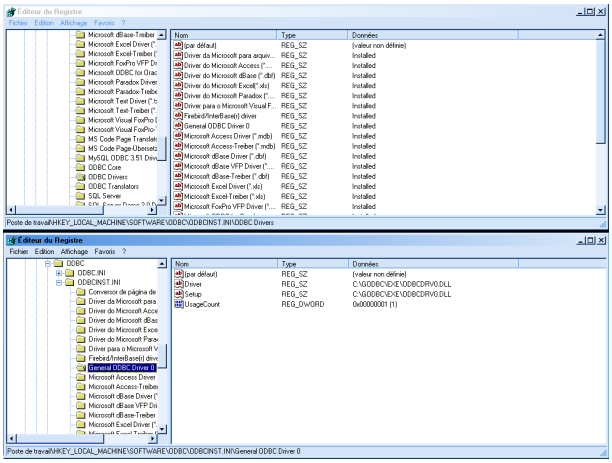


FIG. 7.2 – Base de registre

Lors de la vérification de son installation, nous remarquons qu'il a été impossible pour le système d'exploitation de créer le DNS.

Nous en concluons donc que l'inscription donnée par l'auteur ne permet pas d'installer le pilote fourni.

Récapitulatif

Après ces essais, nous pouvons conclure que le pilote pré-compilé fourni n'est fonctionnel pour l'utilisation que nous souhaitons en faire, alors que son auteur précise bien que celui-ci est censé fonctionné correctement.

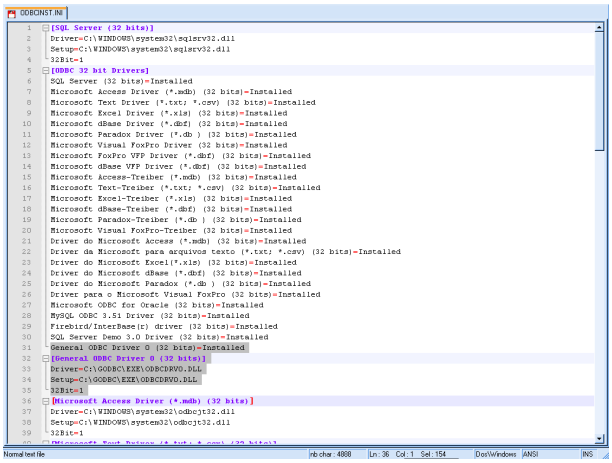

ODBCINST.INI
[SQL Server (32 bits)]
Driver=C:\WINDOWS\system32\sqlsrv32.dll
Setup=C:\WINDOWS\system32\sqlsrv32.dll
32Bit=1
[ODBC 32 bit Drivers]
SQL Server (32 bits)=Installed
Microsoft Access Driver (*.mdb) (32 bits)=Installed
Microsoft Text Driver (*.txt; *.csv) (32 bits)=Installed
Microsoft Excel Driver (*.xls) (32 bits)=Installed
Microsoft dBase Driver (*.dbf) (32 bits)=Installed
Microsoft Paradox Driver (*.db) (32 bits)=Installed
Microsoft Visual FoxPro Driver (32 bits)=Installed
Microsoft FoxPro VFP Driver (*.dbf) (32 bits)=Installed
Microsoft dBase VFP Driver (*.dbf) (32 bits)=Installed
Microsoft Access-Treiber (*.mdb) (32 bits)=Installed
Microsoft Text-Treiber (*.txt; *.csv) (32 bits)=Installed
Microsoft Excel-Treiber (*.xls) (32 bits)=Installed
Microsoft dBase-Treiber (*.dbf) (32 bits)=Installed
Microsoft Paradox-Treiber (*.db) (32 bits)=Installed
Microsoft Visual FoxPro-Treiber (32 bits)=Installed
Driver do Microsoft Access (*.mdb) (32 bits)=Installed
Driver da Microsoft para arquivos texto (*.txt; *.csv) (32 bits)=Installed
Driver do Microsoft Excel(*.xls) (32 bits)=Installed
Driver do Microsoft dBase (*.dbf) (32 bits)=Installed
Driver do Microsoft Paradox (*.db) (32 bits)=Installed
Driver para o Microsoft Visual FoxPro (32 bits)=Installed
Microsoft ODBC for Oracle (32 bits)=Installed
MySQL ODBC 3.51 Driver (32 bits)=Installed
Firebird/InterBase(r) driver (32 bits)=Installed
SQL Server Demo 3.0 Driver (32 bits)=Installed
General ODBC Driver 0 (32 bits)=Installed
[General ODBC Driver 0 (32 bits)]
Driver=C:\GODBC\EXE\ODBCDRV0.DLL
Setup=C:\GODBC\EXE\ODBCDRV0.DLL
32Bit=1
[Microsoft Access Driver (*.mdb) (32 bits)]
Driver=C:\WINDOWS\system32\odbcjt32.dll
Setup=C:\WINDOWS\system32\odbcjt32.dll
32Bit=1

FIG. 7.3 – ODBC initialisation

7.2 Utilisation du code fourni et d'outils supplémentaires

Étant que le pilote pré-compilé fourni ne semble pas fonctionner, nous avons donc choisi d'utiliser le code source fourni pour l'améliorer et l'adapter à nos besoins.

Pour pouvoir développer ce code, nous avons choisi d'utiliser comme compilateur *Dev-C++*.

La première compilation nous a permis de comprendre que de nombreuses bibliothèques étaient nécessaires pour pouvoir compiler le pilote.

Nous sommes parvenus à trouver une plateforme de bibliothèques de *Microsoft* : le *SDK ODBC*. Cette plateforme nous permet de disposer de nombreuses bibliothèques de fonctions, qui seront souvent utilisées par la suite.

Après étude des fonctions disponibles grâce au *SDK ODBC*, nous avons pu compiler le code source fourni. Mais, lors du passage par l'administrateur ODBC, il s'est avéré que la création d'un DNS, assurant l'utilisation du pilote, était toujours impossible et engendrait l'affichage des mêmes erreurs que précédemment.

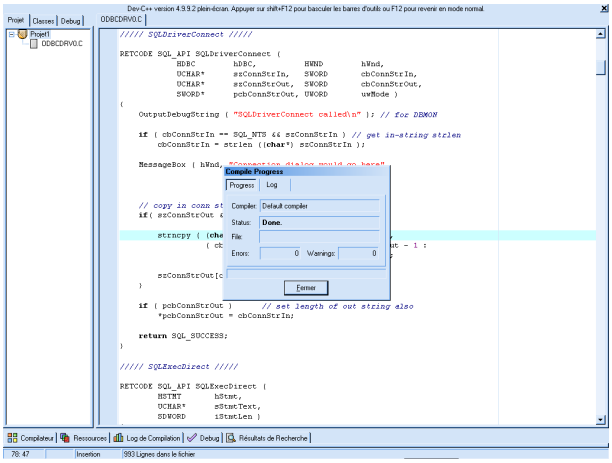


FIG. 7.4 – Compilation

7.2.1 Affichage des avertissements

Après ces vérifications faites depuis l'administrateur ODBC, nous avons commencé une étude plus approfondie du code source du pilote ODBC fourni.

Nous avons constaté que l'auteur de ce squelette avait placé, à l'intérieur du code source, plusieurs messages d'avertissements permettant de suivre l'exécution du pilote par l'administrateur ODBC.

Après de nombreux tests, notamment grâce à l'ajout de plusieurs points de contrôle, nous avons constaté que ceux-ci ne s'affichaient tout simplement pas.

L'erreur de ce problème était due à la mauvaise casse appliquée au fichier compilé. La résolution de ce problème s'est faite par le changement de nom du fichier compilé.

7.2.2 Problème lors de la création du DNS

De nombreux problèmes se sont posés à nous lors des tentatives d'installation du pilote depuis l'administrateur ODBC.

Pour tenter de résoudre ces problèmes, nous avons tenté d'utiliser plusieurs solutions possibles.

Recherche d'informations à la source

Afin de pouvoir résoudre les problèmes de compilation que nous avons rencontré, nous avons contacté l'auteur du pilote ODBC grâce à l'adresse *e-mail* fournie sur sa page Web.

Nous lui avons successivement envoyé deux messages, sur ce modèle :

```
Mr Vikash Agarwal,


I have already written you a message to ask you help about the
development of an ODBC driver.
I have kept trying to make it working, however the result is
still the same. Here is a copy of my previous mail :

-----------------------------------------------------------------
After reading the article following article :
http://www.ddj.com/windows/184416434?pgno=1 , I tried to use the
GODBC.dll skeleton driver you provided.

I followed instructions of installation step by step. The driver
appears in the windows ODBC DNS administrator, but an error occures
when I try to create a new DNS. I ask for your help to specify
what is wrong.

Here is the translation of the error message occuring (in French
on my computer) : "Installation routines of "General ODBC Driver 0"
ODBC driver are not reachable. Please re-do the installation of
the ODBC driver."
-----------------------------------------------------------------

[...]
```

Jusqu'à maintenant, le destinataire n'a apporté aucune réponse à ces messages.

Création manuelle d'un DLL

Après avoir rencontré ces nombreuses erreurs de compilation, nous avons cherché à mieux comprendre comment fonctionne un fichier DLL.

Pour cela, nous avons choisi de créer, par nos propres moyens, une bibliothèque de fonctions en utilisant un compilateur de code source, tel que *Dev-C++*.

Ce compilateur propose l'option de créer un fichier DLL, en fournissant immédiatement un squelette permettant de répondre à la création d'un tel fichier. Nous nous devons ensuite de compiler ce squelette par un ensemble de fonctions permettant de construire pleinement la bibliothèque nécessaire à

La création de ce fichier DLL, ainsi que l'étude des fonctions proposées par défaut par le compilateur, nous a permis de vérifier que notre manière de créer et programmer un tel programme était correcte.

Utilisation de points de contrôle

Une première solution nous permettant de comprendre l'échec de l'installation du driver, est de constater l'état du comportement du pilote lorsqu'on fait appel à lui.

Pour cela, nous utilisons des méthodes assez courantes qui permettent, d'une manière relativement aisée, de « pister » l'utilisation des différentes parties du programme (fonctions, conditionnelles, multi-conditionneles, boucles, ...).

Nous pouvons y arriver en insérant des points de contrôle à quelques points stratégiques du programme, notamment à l'entrée de chacune de ses fonctions.

Premièrement, à chaque point de contrôle, nous choisissons d'écrire dans un fichier texte (que l'on appelle communément « fichier de *log* »), un message contenant par exemple une description de ce point de contrôle. Cette méthode est plus intéressante lorsque ceux-ci sont nombreux ou que leur ordre est important.

Deuxièmement, à chaque point de contrôle, nous choisissons d'afficher une boîte de dialogue contenant par exemple une description de ce point de contrôle. Cette méthode est plus intéressante lorsque ceux-ci sont peu nombreux.

Cela permet de suivre l'ensemble des points de contrôle traversés par l'exécution du programme, et donc de déterminer les fonctions qui sont appelées et celles qui ne le sont pas.

Dans notre cas, nous constatons qu'aucune fonction, si ce n'est la fonction

principale `DllMain`, n'est appelée.

Compilation sous un autre IDE

Étant donné que la compilation sous *Dev-C++* se déroule sans encombre, nous nous permettons de penser que le compilateur pourrait être en cause lors de l'échec de l'importation du pilote depuis l'administrateur ODBC.

En effet, de nombreuses options de compilation qui doivent être choisies dans le logiciel, laissent à penser que le pilote compilé ne serait pas compilé de la façon que l'administrateur s'attend à rencontrer.

Nous tentons donc la compilation du pilote sous d'autres compilateur permettant de gérer le *C/C++*, tels qu'*Eclipse* ou *Visual Studio*.

Malheureusement, les compilations sous ces deux logiciels s'avérèrent être plus compliquées que celle sous *Dev-C++*.

De plus, à l'importation du pilote avec l'administrateur ODBC, aucune amélioration dans les résultats ne s'est faite ressentir.

7.2.3 Problème d'appel de fonctions

Après avoir testé la compilation du pilote ODBC avec plusieurs programmes différents, nous ne pouvons que constater le même résultat : le pilote DLL est bien ajouté dans la base de registre *Windows*, car l'administrateur ODBC peut le « voir » sans autre manipulation.

On peut également normalement supprimer du registre, le pilote ODBC installé.

Mais le même problème persiste : on ne peut pas ajouter de DNS à l'aide de l'administrateur ODBC.

Les messages d'erreurs affichés par le système d'exploitation lors de l'installation du pilote ODBC depuis l'administrateur ODBC semblent expliquer le problème par le fait que les fonctions de l'outil ne soient pas accessibles.

Nous avons donc cherché à résoudre ce problème.

Définition de fonctions

Grâce aux études faites sur les bibliothèques de fonctions DLL et les sources fournies dans le *SDK ODBC*, nous avons pu comprendre comment

doivent être définies les fonctions, afin d'espérer obtenir de meilleurs résultats.

Nous avons constaté que la définition des fonctions devait s'accompagner d'une définition supplémentaire, afin de rajouter des informations utiles à la compilation puis à l'exécution du pilote.

Ces informations sont contenues dans un fichier de définition, communément nommé *fichier* `def`.

L'utilisation d'un tel fichier est en fait une technique standard de *Microsoft* pour exporter les fonctions d'un fichier `DLL` lors de l'utilisation de cette bibliothèque par injection (notamment pour l'utilisation que nous en faisons).

En effet, pour que les fonctions soient reconnues, elles doivent toutes être déclarées dans un `.def`. Nous avons donc compilé le pilote avec ce fichier, mais la compilation s'est avérée infructueuse.

Après avoir approfondi l'étude d'un tel mode de programmation, nous avons dû remplacer ce fichier `.def` par une déclaration de fonctions à l'intérieur même de notre programme, en complément à l'implémentation de ces mêmes fonctions. Cette définition de fonctions se fait en donnant le nom de la fonction, ainsi que la taille des arguments qu'elle prend en paramètres.

Ainsi, la fonction principale de notre pilote (qui est appelée à l'exécution) se déclare de cette façon :

```
#pragma comment(linker, "/export:SQLAllocEnv=_SQLAllocEnv@4")
```

Cette façon de définir des fonctions n'est pas évidente au premier abord. Son utilisation n'a pas permis de régler le problème d'accès aux fonctions.

Déclaration de fonctions

Après avoir vérifié la définition de fonction, nous nous sommes intéressés à la déclaration et à la façon d'implémenter ces fonctions.

En effet, afin de savoir comment développer d'une manière correcte, et utiliser les fonctions qui nous sont proposées, nous devons prendre suffisamment de recul et avoir à notre disposition des exemples de manières différentes de programmer.

Pour cela, nous avons télécharger et étudié d'autres codes source de pilotes ODBC existants (tels que *PostGre* et *mySQL*) notamment en comparant le squelette de notre pilote avec le squelette de ces pilotes.

Cette étude nous a permis de mieux cerner la difficulté de programmation d'une telle bibliothèque *DLL* de fonctions.

Options de compilation

En observant les méthodes de réalisation employées dans la création du squelette du pilote ODBC (et également en comparant avec d'autres squelettes de pilote ODBC), nous avons pu déduire que le code source est fait de base avec *Visual Studio*.

Nous avons cherché à savoir pourquoi la compilation de notre pilote se faisait différemment sous l'un ou l'autre des compilateurs disponibles (notamment *Dev-C++*, *Visual Studio* et *Eclipse*).

Nous avons dû commencer par une étude des moyens de compilation d'un tel programme, recherche qui s'est avérée être relativement longue.

Nous avons tout d'abord étudié plus précisément les informations données par le créateur du pilote ODBC sur sa page Internet. Il précise que les fonctions déclarées doivent être à la norme `stdcall`. Après vérification, nous pouvons constater que cette norme est pourtant bien activée lors de la compilation.

Nous avons ensuite recherché de plus précises informations dans les *makefile* (fichiers de compilation automatique) des pilotes précédemment cités. L'étude de ces fichiers, très peu commentés et extrêment difficiles à comprendre, s'est faite très difficilement.

Récapitulatif des résultats obtenus

Cette suite d'études et de vérifications d'erreurs nous a permis de mieux comprendre la création d'un tel pilote.

Ce processus correspond en fait à l'évolution de la réalisation de notre pilote.

Chapitre 8

Solutions envisagées

Suite aux problèmes rencontrés sur une étape aussi critique que celle de la réutilisation du pilote squelette, notre projet s'est soldé par un échec. Nous souhaitons néanmoins que cette expérience, ainsi que les connaissances acquises tout au long du projet, puissent aider à la prise de décision quant au futur du développement du logiciel de M. Guérin.

8.1 Côté contraignant d'ODBC

Nous allons maintenant expliquer la technologie choisie, qui est ODBC, ne peut finalement pas résoudre à notre projet.

8.1.1 Un manque d'informations

La majorité des logiciels faisant appel à ODBC utilisent cette technologie afin d'importer des données depuis une source de données. Les documents et les exemples sur l'application d'ODBC à ce type de besoin sont particulièrement nombreux.

En revanche, les logiciels cherchant à mettre leurs données à disposition par le biais d'ODBC sont rares. De part ce fait, il est difficile de trouver une documentation sur la conception de pilotes ODBC ainsi que sur leurs implémentations dans un logiciel source de données.

La communauté Internet autour du développement de pilote ODBC est très peu active. Nous avons à plusieurs reprises recherché son soutien sur des forums de programmations ainsi qu'auprès d'un programmeur expérimenté. Toutes nos demandes se sont révélées infructueuses.

8.1.2 Une technologie difficile à implémenter

La complexité de l'API ODBC est un autre facteur expliquant le fait que la création de pilotes n'est pas plus répandue. Cette API est composée de 75 fonctions, parmi lesquelles un minimum d'une trentaine d'entre elles doivent être implémentées pour constituer le corps d'un pilote fonctionnel.

Bien que parfois simples, ces fonctions, devant gérer un grand nombre d'exceptions, peuvent être particulièrement lourdes. De plus, il est nécessaire de bien comprendre l'enchainement entre les fonctions pour savoir quelles données seront transmises à une fonction et comment les traiter.

Si nous avions pu commencer la réalisation du pilote, l'implémentation de ces fonctions aurait été une difficulté de taille à surmonter.

8.2 Solutions envisageables pour l'avenir

Afin de parer à l'absence de résultat pour notre projet, nous avons envisagé d'autres solutions possibles.

8.2.1 Poursuivre avec ODBC

Malgré les problèmes posés par un développement s'appuyant sur l'interface ODBC, nous ne pensons pas que la solution du pilote ODBC soit à exclure. Bien sûr, faire fonctionner le pilote squelette ODBC reste une étape vitale pour l'avancement du projet. En continuant dans cette voie, nous pouvons espérer que le problème qui nous a bloqué aurait fini par être résolu ou contourné.

Trouver un soutien

Comme vu précédemment, la technologie ODBC est assez complexe. Pour résoudre les différents problèmes, présents et futurs, il faudra très certainement faire appel à des personnes qualifiées sur la technologie ODBC.

Si nous avons cherché à trouver une aide extérieure, nous pouvons critiquer notre travail.

La communauté vers laquelle nous nous sommes dirigés, bien qu'active et composée de programmeurs expérimentés, n'était pas particulièrement à même de répondre à des questions pointues sur ODBC. Chercher de l'aide

auprès de forums de discussion de Microsoft aurait été plus judicieux. On pourrait espérer alors recevoir de l'aide d'un expert ou d'être aiguillé vers des documentations pertinentes.

Une autre possibilité pour trouver un soutien expérimenté sur ODBC serait de contacter les personnes ayant participé à la création des pilotes ODBC de logiciels open-source tels que MySQL, FireBird ou PostGre.

En disposant d'un soutien suffisant, nous pensons que le projet de pilote ODBC est tout à fait réalisable.

Compléter le pilote squelette

Comme vu précédemment, les fonctions de l'API à implémenter pour réaliser le corps du pilote ODBC peuvent être particulièrement complexes.

La phase de réalisation du pilote doit passer par une étude du fonctionnement interne d'ODBC ainsi que des différentes notions ODBC : « Environnement », « Handle », « Row Cursor », « Statement ».

Ces connaissances sont nécessaires pour comprendre les paramètres et les valeurs de retour de chaque fonction. La documentation du MSDN sur ODBC et son API, relativement bien construite, devrait suffir pour ces recherches.

Si les informations du MSDN sont assez complètes pour permettre une bonne compréhension de chaque fonction, elles ne permettent pas de visualiser la forme de leurs contenus. Pour implémenter correctement une fonction, nous opterions pour s'inspirer de cette même fonction écrite d'autres pilotes ODBC déjà existants. Une comparaison de l'implémentation des fonctions dans différents pilotes permettra de réaliser un code propre, fiable et gérant un maximum d'exceptions.

8.2.2 Utiliser un pilote ODBC déjà existant

En s'appuyant sur nos connaissances actuelles d'ODBC, nous pensons qu'il est possible de réaliser le projet en se basant sur un pilote fonctionnel existant. L'astuce serait d'effectuer la connexion avec un autre logiciel que celui pour lequel le pilote est initialement prévu.

Effectivement, la connexion du pilote passe par l'appel de fonctions identifiables. De plus, nos observations sur différents pilotes nous permettent

d'affirmer que le pilote est globalement indépendant du reste du logiciel avec lequel il travaille.

Suite à une phase d'aprentissage sur les échanges d'informations entre le logiciel d'origine et son pilote, il pourrait donc être possible de simuler cette connexion depuis le logiciel voulu pour utiliser ce même pilote.

Pour comprendre le fonctionnement de la connection, l'administrateur ODBC dispose d'un outil de traçage qu'il serait intéressant d'utiliser. Cet outil créé un journal recensant les appels aux pilotes ODBC installés.

Voici le journal généré lors de l'exportation d'un fichier CVS sous Excel :

```
[...]
EXCEL           b0c-1038 EXIT  SQLDriverConnectW  with return code 0
                                                              (SQL_SUCCESS)
HDBC                 01C91D60
HWND                 00270436
WCHAR *              0x6CD8B1C0 [      -3] <Invalid string length!>
SWORD                        -3
WCHAR *              0x6CD8B1C0
SWORD                        -3
SWORD *              0x00000000
UWORD                         3 <SQL_DRIVER_COMPLETE_REQUIRED>

EXCEL           b0c-1038 ENTER SQLGetInfoW
HDBC                 01C91D60
UWORD                        77 <SQL_DRIVER_ODBC_VER>
PTR                  0x001BDC18
SWORD                        24
SWORD *              0x00000000
[...]
```

8.2.3 Utilisation d'une autre API

Ayant déjà exposé les qualités et les défauts des autres technologies qui permettent d'effectuer un pilote d'echange de données, nous ne nous étendrons pas sur cette solution. Nous rappelons tout de même que *OLE DB*, proche d'ODBC et compatible avec un grand nombre de SGBD du marché, est un candidat potentiel pour la réalisation d'un pilote.

Si le résultat final serait approximativement le même, nous craignons que les problèmes rencontrés avec ODBC soient encore plus prononcés pour

d'autres API moins populaires.

8.2.4 Utiliser un fichier de données intermédiaire

Notre dernière solution proposée est la réalisation d'un exportateur de données enregistrant les données dans un fichier sous un format reconnu par les SGBD du marché.

Pour cette solution, le processus d'échange d'information est effectué en 2 temps :

1. L'utilisateur aurait à exporter des données depuis le logiciel source de données vers un fichier.
2. L'utilisateur indiquerait alors au SGBD le fichier à importer pour en extraire les données.

Etant lu par certains pilotes ODBC, le format CSV (Coma Separated Values) pourrait être traité par les modules d'importation des SGBD du marché. Il se présente comme un excellent candidat.

Cette solution a le désavantage par rapport à ODBC de rendre l'échange de données plus long et plus compliqué pour l'utilisateur. En revenche, elle permet de conserver des données sous forme de fichier importable dans n'importe quel SGBD. Aussi, cette solution, techniquement simple, ne pose pas de problème de recouvrement de transaction qu'il faudrait gérer avec un pilote ODBC.

8.3 Développement d'une solution : l'exportation de données au format CSV

Parmi toutes les solutions envisagées, nous pensons que celle proposant la réalisation d'un exportateur de données au format CSV, à la fois simple et efficace, est la plus apropriée pour le futur du développement du projet.

Nous allons donc nous intéresser de plus près à celle-ci en reprenant brièvement les différentes étapes de gestion d'un projet, c'est-à-dire, de bibliographie et de conception.

8.3.1 Cahier des charges

Nous ne referons pas ici, une bibliographie complète du projet, celui-ci restant sensiblement le même. Nous exposerons en revanche les notions nouvelles et les changement importants dans le projet.

Objectifs

L'objectif du projet reste de permettre au logiciel de M. GUÉRIN d'échanger des données avec un SGBD du marché. Cet objectif est atteint indirectement en ajoutant une fonctionnalité d'exportation de donnée au format CSV à ce logiciel.

Ce fichier pourra être envoyé à un SGBD en utilisant un pilote ODBC déjà existant.

Contraintes et besoins du projet (FIG. 8.1)

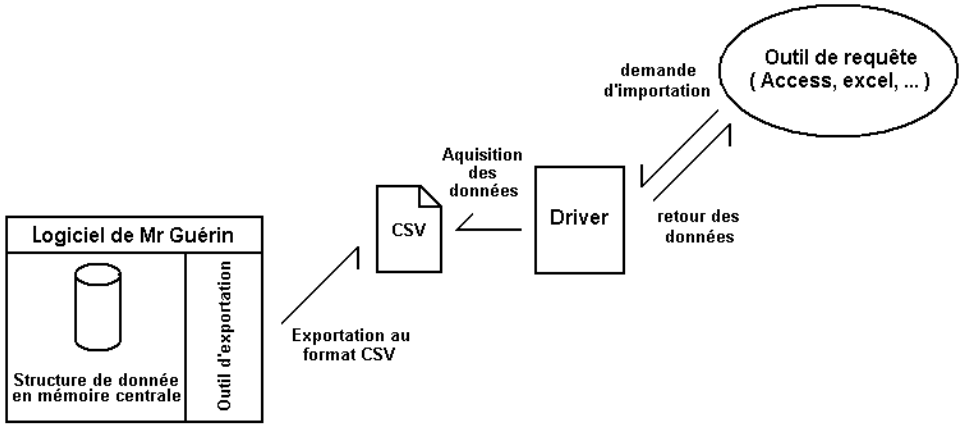


FIG. 8.1 – Schéma général

La brique logiciel développée sera intégrée au logiciel de M. GUÉRIN. Ce dernier étant développé en Java, le travail réalisé doit être développé en Java. Aussi, cet outil devant être facile à implémenter, il se présentera sous la forme d'un « package » d'objets.

Un objet de ce dernier disposera d'une interface simple et bien documentée, qui permettra la communication avec le reste du logiciel.

La source de données depuis laquelle les données seront extraites, est située en mémoire centrale et est connue uniquement par le logiciel de M. Guérin. Ce logiciel devra transmettre la position de ces données en mémoire centrale ou une copie de ces donnée via l'interface du *package* développé.

L'outil développé devra permettre de sélectionner une partie des données parmi celles qui lui seront transmises.

Un système de pseudo-requête sera à mettre au point.

Le logiciel de M. Guérin étant destiné a être diffusé, l'outil devra implémenter une IHM, et être le plus ergonomique possible à utiliser.

L'IHM sera très certainement réalisée avec *Swing*, la bibliothèque graphique la plus utilisée en Java.

Division du travail en événement métier

En s'appuyant sur les contraintes exposées ci-dessus, voici les différentes actions que devra être capable d'effectuer le travail réalisé.

1. À la demande du logiciel de M. Guérin, l'outil doit ouvrir une fenêtre disposant d'une interface permettant à l'utilisateur de choisir les données à exporter.
2. L'utilisateur créé une requête simple. Celle-ci est analysée afin de déterminer les données à extraire
3. Les données sont extraites et écrites dans un fichier au format CSV
4. La fenêtre de l'outil se ferme et rend la main au reste du logiciel de M. Guérin

Cette suite d'action peut être illustrée par un diagramme de cas d'utilisation :

8.3.2 Complément bibliographique

Cette partie a pour rôle d'apporter des informations synthétiques utiles à la conception de l'exportateur.

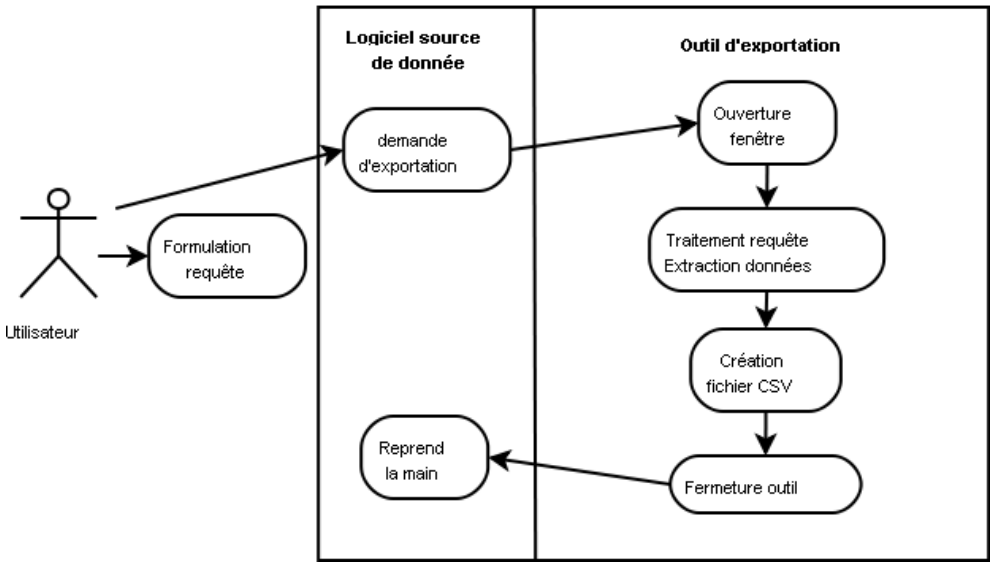


FIG. 8.2 – Diagramme de cas d'utilisation

Fichier CSV

Un fichier CSV (Comma-Separated Values) un format non-propriétaire de fichier texte, généralement enregistré avec l'extension `.csv`, contenant des valeurs structurées à la manière d'une table de données.

Chaque ligne du fichier correspond à une ligne du tableau, la 1re étant généralement utilisée pour contenir le nom des colonnes. Les valeurs d'une même ligne sont séparées par un caractère non-standardisé, généralement « ,fg (« ;fg pour certaines localités dont la France).

Quelques tests nous ont montré qu'un mauvais choix de séparateur pouvait compromettre une bonne exportation via ODBC.

CSV est un format essentiellement utilisé par Excel qui ne permet pas initialement de contenir plusieurs tables dans un même fichier.

En revanche, il doit sa popularité à sa simplicité et sa génération facile. Il est entre autre utilisé pour effectuer des échanges de données entre SGBD n'ayant pas d'autre format de fichier en commun.

Voici quelques règles supplémentaires utilisées pour écrire un fichier CSV :
- La valeur d'un champ peut-être entourée de guillemet
- Un guillemet apparaissant dans la valeur d'un champ doit être doublé
- Un caractère séparateur apparaissant dans une valeur entourée de guille-

ment ne sera pas considéré comme séparateur

Attention ! Ces règles risquent de ne pas être supportées par le pilote ODBC choisi pour réaliser le projet.

Java.Swing

Swing est une bibliothèque graphique Java faisant partie du package *Java Foundation Classes*, qui constitue une API riche pour la conception d'IHM. Il permet de construire des interfaces sophistiquées à partir de composants imbriqués.

S'imposant comme une évolution majeure apportée par Java 2 et ayant pour but de remplacer AWT, Swing est plus léger, mieux structuré et plus étendu que son ancêtre. Swing se démarque également par sa portabilité et son apparence indépendante de la plateforme.

Swing est très ancré dans le modèle MVC. Cela lui permet, entre autres, de disposer d'une apparence facilement modifiable et d'être parfaitement adapté au modèle Observer/Observable.

8.3.3 Conception

Afin de pouvoir réaliser le dossier de conception, nous devons d'abord définir la vue générale de l'architecture du projet, ainsi que la conception des différents éléments.

(A) Vue générale de l'architecture du projet

Les éléments formant le contexte du projet sont les suivants :
- Le SGBD client
- L'administrateur de source de données ODBC
- Le gestionnaire de pilote ODBC
- L'application source de données, à laquelle il faut rajouter la fonctionnalité d'exportation au format CSV

Nous ne décrirons pas ici ces éléments ici, cela ayant déjà été fait dans le dossier de conception (voir FIG. 8.3).

Les éléments à réaliser sont les suivants :

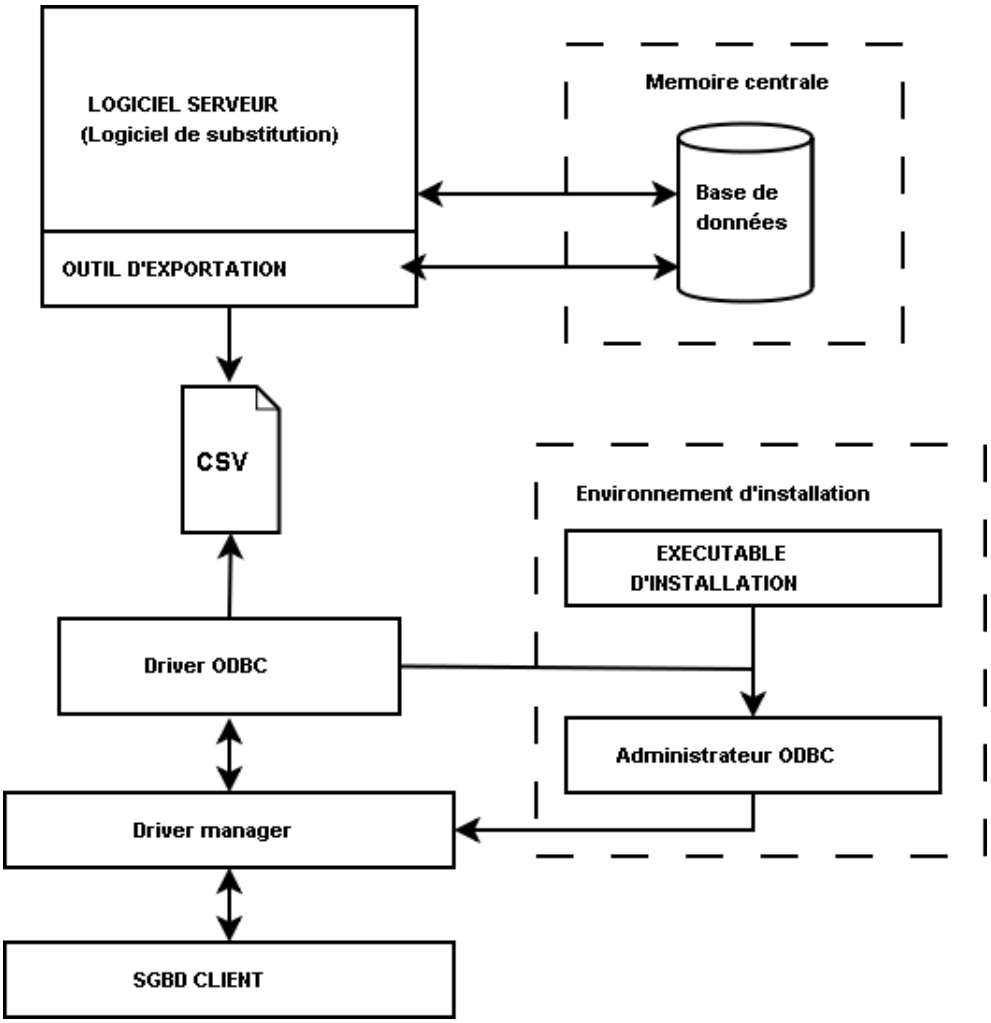


FIG. 8.3 – Schéma de l'ensemble fonctionnel

Le package d'exportation de données :

L'implémentation de cet élément représente la plus grande partie de la réalisation du projet. Ce package a diverses fonctionnalités permettant d'aboutir à l'exportation de données au format CSV.

Il se découpe naturellement en quatres parties ayant chacune un rôle précis :

L'interface utilisateur constituant la vue du projet :

Sa réalisation passera par la création d'une IHM aussi simple que possible pour l'utilisateur. Cette partie est également chargée de réagir aux actions de l'utilisateur.

L'extracteur de données :

Il a pour rôle de récupérer la requête pseudo-SQL simple envoyée par l'utilisateur. Après l'avoir analysée, il doit sélectionner et renvoyer les données correspondantes.

Le convertisseur de données :

Cet élément doit être capable de convertir les données extraites sous forme de fichier CSV, et de l'enregistrer dans un dossier

L'interface d'exportation :

Composé essentiellement des fonctions interfaçant notre projet avec le reste du programme, cet objet devra également se charger de l'initialisation et de la clôture de chaque exportation.

L'exécutable d'installation :

Cet outil doit être utilisé lors de l'installation du logiciel source de données ; et a pour rôle d'automatiser l'installation du pilote ODBC et la création de son DNS.

Bien que le pilote ODBC permettant l'importation de fichier CSV soit présent sur toutes les plateformes Windows, nous trouvons plus prudent de réimplanter un pilote testé que l'on sait fonctionnel.

La création du DNS effectuée pendant l'installation du logiciel indiquera au pilote la position du dossier contenant les fichiers CSV. Ainsi, l'utilisateur n'aura ni à toucher à l'administrateur ODBC, ni a s'occuper de la localisation des fichiers CSV exportés.

Le logiciel de substitution :

Le logiciel de M. Guérin étant encore en cours de développement, un logiciel de substitution devra être réalisé. Celui-ci disposera d'une base

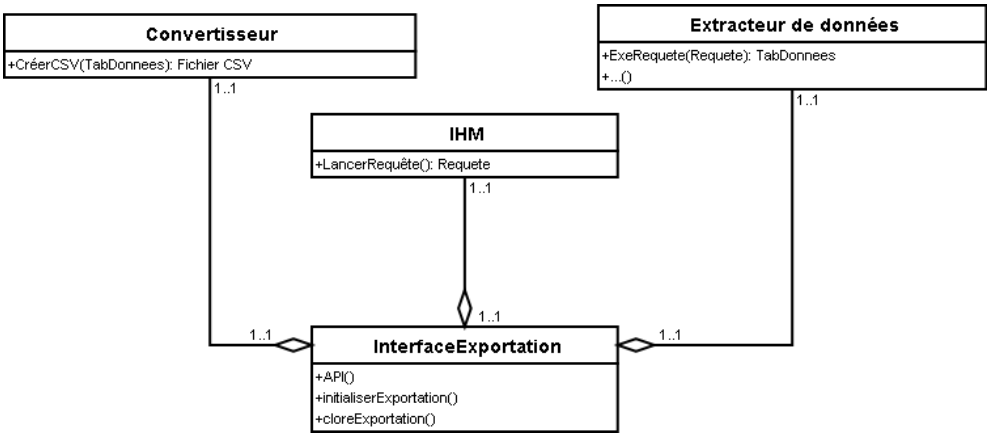


FIG. 8.4 – Diagramme de classes

de données qui sera envoyé à l'outil d'exportation par l'interface de ce dernier.

(B) Conception des différents éléments

Nous n'étudierons ici pas toutes les parties en détail comme cela avait pu être effectué lors du dossier de conception. Nous évoquerons seulement les outils utilisés et les méthodes mises en œuvre pour chaque partie.

Package d'exportation de données (FIG. 8.4) :

- Méthode de conception :
  Le package d'exportation qui sera implanté dans le logiciel source de donnée doit entièrement être écrit en Java. Chaque partie décrite précédemment composant ce bloc constituera un objet du package.

Interface utilisateur constituant la vue du projet :

- Outils et bibliothèques utilisées :
  Bibliothèque graphique Java : Classes du package `javax.swing`
- Méthode de conception :
  Pour rendre la construction de requête simple et pour éviter les requêtes invalides, celle-ci doit être particulièrement guidée. L'interface sera composée de blocs mis à jour automatiquement, permettant à l'utilisateur de sélectionner une table, un ou plusieurs attributs dans cette dernière, puis de poser une ou plusieurs contraintes simples.

Afin de réaliser une interface dynamique, cette partie sera implémentée selon le modèle MVC, en choisissant comme modèle la requête et comme vue l'interface Swing. Cette classe a une accessibilité restreinte au package.

Extracteur de données :

- Méthode de conception :
  Cet objet ne pose aucune difficulté particulière. Le choix du type de structure de données envoyée à l'outil d'exportation est néanmoins nécessaire pour réaliser cette partie. Cette classe a une accessibilité restreinte au package.

Convertisseur de données :

- Outils et bibliothèques utilisées :
  Classes de lecture/écriture de fichiers : Classes du package `java.io.File`
- Méthode de conception :
  Cet objet doit simplement créer les fichiers au format `.csv` avec les données transmises par l'extracteur de données. Après quoi, le fichier doit être enregistré dans un répertoire prédéfini, fixe et relatif à la position du logiciel, afin que le pilote ODBC puisse facilement retrouver les informations. Cette classe a une accessibilité restreinte au package.

L'interface d'exportation :

- Méthode de conception :
  C'est la seule classe publique du package. Cette classe implémente l'API permettant le lien avec le reste de l'application d'une part, et coordonne les différents éléments du package d'autre part. Elle doit donc disposer d'une instance de chaque classe qu'elle se chargera d'initialiser.
  La structure des données reçue au niveau de l'API reste à définir. Pour une table de donnée, celle-ci peut se présenter sous forme de tableau ou sous forme d'une liste chainée de tuples.

L'exécutable d'installation :

Au cours de notre étude sur ODBC, nous avons remarqué que cette partie peut aussi bien être réalisé en Java qu'en C. ODBC étant natif C, et la bibliothèque `odbcinst.h` étant facile à utiliser, nous proposons de réaliser ce module en C.

- Outils et bibliothèques utilisées :
  Bibliothèque ODBC : `odbcinst.h`
- Méthode de conception :
  Pour réaliser cette partie, on pourra repartir de l'outil d'enregistrement de pilote de GODBC. Pour ce qui est de la création d'un DNS, Internet regorge d'exemples et de documentation sur laquelle il est possible de s'appuyer.
  Á son exécution, le exécutable d'installation prendra comme paramètre la position du logiciel source de donnée. Appelé lors de l'installation du logiciel, il pourra alors directement récupérer cette donnée nécessaire pour la création du DNS.

Logiciel de substitution :

La méthode de réalisation du logiciel de substitution au logiciel de M. Guérin est identique à la description faite dans le précédent rapport de conception.

Conclusion sur le développement d'une solution

Parmi toutes les solutions envisagées, nous venons d'étudier celle proposant la réalisation d'un exportateur de données au format CSV. Cette solution, à la fois simple et efficace, est la plus apropriée pour le futur du développement du projet.

Après s'être intéressés à celle-ci (en reprenant brièvement les différentes étapes de gestion d'un projet), nous pouvons penser que cette solution pourrait permettre de respecter les besoins du projet, et de répondre aux exigences qui ont été définies, sans toutefois dénaturer le projet.

Conclusion

Cette conclusion va nous pemettre d'établir un bilan récapitulatif, et de donner une analyse personnelle sur la gestion du projet.

En ce qui concerne l'aspect technique, et après avoir pris du recul, nous constatons que la création d'un pilote ODBC n'est pas la seule solution possible à adopter pour répondre aux exigences de notre projet.

En effet, les problèmes rencontrés au cours de ce projet nous laissent penser qu'ODBC est une solution intéressante mais néanmoins difficile à mettre en place. Toutefois, des solutions alternatives, mais moins audacieuses et ne répondant pas forcément aussi bien au besoin, existent.

Nous pouvons constater que notre travail pourrait être réutilisé en tant que base pour continuer ce projet sur la piste du driver d'ODBC. La continuation de ce projet serait ainsi facilitée, grâce à l'apport d'une étude bibliographique et d'un support de travail consistant.

Enfin, au-delà du pur apprentissage technique, ce projet transversal nous a appris, car c'est son premier objectif, à évaluer et à se confronter aux principes, aux rigueurs et aux difficultés de la tenue et de la gestion d'un projet informatique, activité qui sera très certainement la nôtre après l'obtention du diplôme.

Un petit aparté serait également nécessaire pour dresser une analyse comparative du planning prévisionnel (fourni dans le rapport de bibliographie, puis raffiné dans le dossier de conception) et du planning effectif. En effet, nous n'avons pas su évaluer assez tôt les difficultés qui nous sont apparues. L'apprentissage de la gestion de projet permet d'éviter de commettre de telles erreurs.

Quatrième partie

Dimension commerciale

Chapitre 9

Présentation de l'analyse

9.1 Présentation de l'entreprise

Dans le cadre du projet de lancement d'une start-up, M. Guérin, notre tuteur entreprise, compte développer un logiciel ayant besoin de gérer des données dans une SGBD propriétaire.

9.2 Présentation du projet

L'objectif de ce projet est d'ajouter une fonctionnalité au logiciel développé par M. Guérin. Plus précisément, nous devons réaliser un driver capable d'exporter ces données vers des outils de requêtes du marché (*Business Objects*, *Cognos*, *Access*, *Excel*, ...).

Le driver que nous produirons disposera d'une interface qui rendra notre travail intégrable non seulement au logiciel de M. Guérin, mais également à tout autre projet de développement ayant besoin de pouvoir exporter des données vers des outils de requêtes.

9.3 Problématique

Le produit de notre travail sera rendu libre à la fin de notre projet. Malgré cela, nous réaliserons notre rapport sur l'approche marketing, comme si ce produit avait du être commercialisé.

Problèmatique Quelle serait la place du driver que nous devons réaliser, sur le marché de l'informatique ?

Dans cette étude, nous déterminerons la clientèle potentielle de ce driver. Nous chercherons à étudier les solutions similaires déjà existantes et les logiciels qui les utilisent.

Ce dossier pourra mettre en évidence en quoi ce produit se démarque de ses concurrents, et si sa commercialisation serait possible et viable sur le marché de l'informatique.

Chapitre 10

Analyse commerciale du projet

10.1 Objectifs de l'étude

- Prendre conscience que tout projet informatique a des dimensions non techniques qu'un responsable de projet doit savoir prendre en compte.

- Permettre la mise en oeuvre d'une réflexion de gestion en réel dans le cadre d'une situation concrète : le projet réalisé dans le département INFORMATIQUE, en s'appuyant sur les acquis enseignés dans les différentes disciplines de gestion (analyse organisationnelle, informatique et entreprise d'une part ; et marketing d'autre part).

10.2 Présentation

Dans le cadre d'un projet en collaboration avec une entreprise effectué à Polytech'Nantes, nous devons étudier et développer un outil, appelé « pilote », permettant à des logiciels tels qu'*Excel* et *Access* d'accéder aux données d'un logiciel dans lequel cet outil est implanté.

Ce travail a été réalisé pour M. Guérin, qui souhaite le réutiliser pour concevoir un logiciel dans le cadre du lancement d'une *start-up*.

Le produit de notre travail sera rendu libre à la fin de notre projet. Malgré cela, nous avons réaliser une étude marketing sur ce dernier, comme s'il avait

du être commercialisé.

La problématique
Quelle serait la place du pilote que nous devons réaliser sur le marché de l'informatique ?

Le rapport suivant est composé d'une brève étude de marché, suivie de propositions en stratégie marketing.

Dans l'étude de marché, nous définirons clairement l'intérêt de notre pilote, et déterminerons alors la clientèle potentielle qu'il pourrait atteindre.
Nous étudierons ensuite les produits concurrents, les solutions similaires déjà existantes, en citant pour chacun les logiciels qui les utilisent.
Nous chercherons également à anticiper les évolutions possibles du marché, afin de déterminer la viabilité à long terme de la commercialisation du produit.

Lors de la définition de la stratégie marketing, nous étudierons tout d'abord le coût de développement du produit.
Cette partie contiendra également un choix des modes de distribution les plus adaptés, ainsi que différents services qui pourraient être fournis en complément de notre produit.
Enfin, nous proposerons diverses méthodes de commmunication.

10.3 Réflexion marketing sur l'exploitation commerciale du projet

On se basera sur les grandes étapes de la démarche marketing : l'étude de marché et la définition d'une stratégie commerciale.

10.3.1 Étude de marché

L'objectif de ce projet est d'ajouter une fonctionnalité au logiciel développé par M. Guérin. Plus précisément, nous devons réaliser un pilote capable d'exporter ces données vers des outils de requêtes du marché (*Business Objects*, *Cognos*, *Access*, *Excel*, ...).

Le pilote que nous produirons disposera d'une interface qui rendra notre travail intégrable non seulement au logiciel de M. Guérin, mais également

à tout autre projet de développement ayant besoin de pouvoir exporter des données vers des outils de requêtes.

Offre

Présentation du projet

Le produit que nous réalisons est conçu pour être intégré dans un logiciel au cours de son développement afin d'y ajouter une fonctionnalité.

Une fois correctement implanté et installé, il permettra à des outils de traitement d'information (*Excel*, *Access*,...) ou à des SGBD (*Oracle*, *mySQL*,...) de récupérer des données depuis ce logiciel.

Forces

Sans rentrer dans des détails techniques, nous pouvons souligner que l'interface utilisée pour échanger ces informations, ODBC, est un standard Microsoft utilisé par la grande majorité des SGBD et des outils de traitement d'information du marché. Un logiciel utilisant notre produit pourra donc partager ces informations avec une importante panoplie d'outils.

Notre produit sera capable de s'adapter à tout type de source de données, relationnel ou non. L'utilisateur ne sera donc pas obligé de repenser sa structure de données pour implémenter ce produit.

Le pilote sera capable de faire transiter de lourdes quantités d'informations, interprétables par les SGBD sous forme de tables.

Notre produit implémentera une interface qui la rendra simple à inclure dans un logiciel en développement.

Faiblesses

Notre pilote implémentera un interpréteur SQL simple, qui offrira un choix des données à importer limité.

Malrgé la mise à disposition d'une API simple, une personne s'y connaissant peu en programmation risque de ne pas réussir à inclure ce produit dans son projet.

Exemple d'utilisation du pilote

Pour illustrer l'utilité de notre produit, nous prendrons pour exemple un logiciel d'expérimentation scientifique implémentant ce dernier. Un utilisateur effectue une expérience avec ce logiciel et obtient des résultats qu'il doit analyser. Notre pilote lui offre la possibilité d'importer ces résultats depuis le logiciel d'expérimentation dans *Excel*, outil avec lequel il pourra effectuer l'analyse souhaitée.

Ce même utilisateur réitère cette expérience un très grand nombre de fois et obtient une liste de résultats dans son logiciel d'expérimentation. Il peut transférer ces données dans une SGBD classique en se servant de notre pilote. Il pourra alors utiliser les outils fournis par SQL pour traiter ces données.

Demande

Cette partie permet la définition du besoin cible, et l'adaptation du produit à la cible.

Actuellement, les utilisateurs ne sont pas précisément identifiés.

Nous pouvons tout de même souligner que notre produit s'adresse à une clientèle avec les caractéristiques suivantes :

- Le client développe un logiciel. Notre produit vise donc une clientèle ayant une expérience en programmation
- Ce logiciel retourne des résultats qu'il faut analyser depuis un logciel de traitement de données ou stocker dans une SGBD. Le cadre de cette utilisation est certainement professionnel.

Les SEII et les entreprises de développement de progiciels sont donc clients potentiels de ce produit.

Les progiciels qui en ont le plus besoin produisent un grand nombre de données sur lesquelles on pourrait souhaiter travailler, tels que les logiciels de comptabilité, d'aide à la décision, d'analyses médicales, ...

Ce produit pourrait donc rentrer dans la conception de bon nombre de logiciels.

Ce produit n'est cependant pas un produit d'achat courant. On estimera à 10 le nombre de ventes par an.

Environnement

Concurrents à notre produit

Actuellement sur le marché, il existe plusieurs pilotes réalisant le transfert de données de la même manière que notre produit. Chacun de ces pilotes peut être acheté en ligne, le plus souvent à un prix variant entre 500 et 1200 €.

Notamment, on trouvera les pilotes de *EasySoft* et de *OpenLinkSoftware*.

Démarcation de notre produit par rapport à la concurrence
Notre produit aurait l'avantage d'être relativement simple, assez épuré et sûrement à plus faible coût que ses concurrents.

Faiblesses de notre produit par rapport à la concurrence
Notre produit a le désavantage d'être moins complet.

Anticipation de l'évolution du marché

La concurrence n'évoluera sûrement pas.

Nous pourrons peut-être constater l'apparition de nouveaux pilotes basés sur la technologie ODBC, mais qui auront toujours les mêmes caractéristiques que ceux présentés dernièrement.

En revanche, la technologie (ODBC) permettant l'échange d'informations à l'aide de notre produit risque de se voir remplacer par une autre plus moderne. On notera les tentatives de Microsoft de remplacer cette technologie par une plus récente, telle que *OLE DB* ou *ADO*.

Une autre menace pour notre produit serait l'arrivée dans le domaine du libre d'un produit au rôle similaire, ce qui déclencherait une migration de clients vers ce nouveau produit.

Un moyen de contrer cette évolution serait de proposer des services efficaces d'accompagnement et d'aide à la maintenance.

10.3.2 Stratégie commerciale

Cette partie va nous permettre de définir la stratégie commerciale à respecter lors de la mise sur le marché de notre produit.

Cette stratégie commerciale prend en compte un maximum de paramètres sur le marché du produit à vendre, afin de nous mettre en adéquation avec les exigences explicites ou implicites du marché sur lequel nous agissons.

Avant toute chose, afin de pouvoir lancer la commercialisation du produit, nous avons besoin de déterminer le coût de notre produit.

Une fois le prix connu, nous aurons besoin de choisir un ou plusieurs types de distribution permettant la vente de notre produit la plus en adéquation avec la clientèle ciblée.

La politique de communication, la publicité, la promotion et l'organisation de la vente des produits représentent une grande partie de l'aspect visible du marketing auprès du grand public.

Les bases de cette stratégie sont également de définir les services complémentaires à notre produit.

Etude de coût

Afin de déterminer le coût à appliquer au produit lors de sa mise en vente, nous allons chercher à connaître le prix de revient du produit, puis les marges à appliquer sur ce prix afin d'obtenir le prix de vente final.

Ce prix de vente doit au minimum être supérieur au prix de revient du produit. Afin de connaître ce prix de revient, nous allons étudier quelques paramètres qui nous permettront de définir (ou au moins d'appproximer) le coût réel de revient du produit.

Ce prix devra notamment correspondre au travail effectué par les personnes réalisatrices du projet.

Pour simplifier le travail, nous considèrerons que notre groupe de travail a une efficacité équivalente à 50% de la capacité d'un ingénieur expérimenté. On considère également que ce travail effectué à des fins comerciales nécéssiterait seulement la moitié du travail de bibliogaphie et de modélisation que nous avons réalisé.
Un ingénieur expérimenté aurait donc réalisé ce travail en 120 heures.

Payé 30000€ par an, l'ingénieur aurait donc coûté 2060€ au projet.
Etant donné que l'on a estimé le nombre de ventes à 10 par an, un prix de vente final de 300€ nous permettrait de rentabiliser rapidement le produit.

Type de distribution

Après avoir étudié le coût de notre produit, nous allons déterminer la façon dont le produit sera distribué au client.
Ce type de distribution doit être choisi pour permettre une vente simple, et une réception rapide.

Nous pouvons utiliser le mode de distribution classique : achat par courrier, puis envoi du produit par courrier sécurisé.

Mais ce moyen de distribution n'est pas très moderne et suffisamment efficace.

Un autre type de distribution de notre produit pourrait être la vente et le téléchargement en ligne de celui-ci.
Ce mode de distribution a l'avantage d'être plus efficace, et plus intéressant pour le client.

Services complémentaires

Des services pourraient être proposés en supplément à l'achat de notre produit.

Si notre client a des lacunes en informatique, il risque fortement de rencontrer des difficultés à implémenter notre produit. Afin d'éviter cela, on peut tout d'abord proposer une aide à l'implémentation payante.

Par la suite, dans le cas où l'utilisateur rencontrerait des problèmes avec notre produit, un service de maintenance technique pourra être mis à sa dispotion afin d'assurer le bon fonctionnement du produit.

Communication

Afin de promouvoir notre produit, nous devons choisir un bon mode de communication.

Lors de la précédente étude de marché, nous avons établi que notre clientèle cible sera les SSII et les entreprises de développement de progiciels.
Ces entreprises surveillent les innovations dans le domaine de l'informatique. Pour les atteindre, nous pourrons présenter notre produit dans un article de magazine spécialisé dans le développement informatique.

Lorsqu'un développeur est confronté à des difficultés, il se dirige principalement vers des moteurs de recherche Internet.
Une communication intéressante et efficace serait de faire appel à des entreprises de référencement afin d'être mieux représentés sur ces moteurs de recherche. L'exposition à la clientèle serait alors plus importante.

Un dernier moyen d'atteindre les entreprises ciblées serait de les tenir informées de l'existence de notre produit, de ses avantages et des services proposés.

Conclusion

L'étude commerciale de notre produit nous a permis d'étudier le marché et de définir une stratégie commerciale qui conviendrait à la diffusion de notre produit.

Cette étude permettra donc d'optimiser la vente de notre produit.

Annexes

Etude bibliographique

Objectif

Ce document a pour but de faire le point un mois après le debut du projet transversal. Vous y trouverez une introduction sur le travail demandé, une description de cette demande, le déroulement du projet étape par étape ainsi qu'une bibliographie.

Introduction

Entrant dans le cadre du lancement d'une nouvelle entreprise, le projet qui nous a été confié a pour objectif la réalisation d'un outil informatique sur mesure.

La finalité de notre travail est le développement d'un driver permettant la communication entre un SGBD propriétaire de notre tuteur entreprise et des SGBD classiques (*i.e. Access*). Grâce à un tel driver, notre tuteur souhaite rendre son produit compatible avec les logiciels du marché, et de le publier auprès de sa clientèle.

Description de la demande

Le driver que nous fournirons doit utiliser l'un des protocoles de communications mis à disposition par la majorité des outils de requête du marché tel que ODBC, OLE, ...

Les bases de cet outil pouvant être longues a développer, nous nous appuierons sur un projet déjà existant dans le domaine du libre traitant des fichiers plats ou des structures de BD simples connues. Après en avoir étudié les sources, nous modifierons ce driver en substituant la couche de plus haut niveau, traitant la structure de BD connue, par une couche d'interface facilement utilisable. Si le projet avance correctement, une étape supplémen-

taire pourrait être de finir la jonction entre le driver et le logiciel SGBD propriétaire de notre tuteur d'entreprise.

Le produit rendu sera sous la forme d'un *DLL* à inclure dans les sources du programme pour lequel il est destiné. Nous remettrons les sources commentées du driver, développées en java sous Eclipse. Nous y joindrons une documentation technique ainsi que la synthèse de notre recherche sur la licence GNU. Pour prouver la qualité de notre travail, nous devrons montrer le bon fonctionnement et la simplicité d'utilisation du driver sur des fichiers plats.

Déroulement du projet

Etape nř1 : Bibliographie et analyse des besoins

Du : 1 octobre 2007
Au : 7 décembre 2007

Nous commencerons cette étape par l'étude des différents protocoles de communication des SGBD, et nous affinerons nos recherches sur le protocole candidat pour la suite du projet. Nous rechercherons également un driver existant facilement analysable et libre de droit.
Enfin, nous produirons une documentation sur la licence GNU.
Toute autre documentation nécessaire pour la suite du projet sera bien entendu ajoutée à la bibliographie.
Le travail rendu au terme de cette première étape comprendra :

- le cahier des charges ci-joint
- une bibliographie complète
- un planning de gestion de projet
- des diagrammes UML décrivant de manière générale le driver à modifier.

Etape nř2 : Conception

Du : 8 décembre 2007
Au : 15 février 2008

Cette partie a pour objectif de définir précisément le travail que nous effectuerons dans la étape de conception. Cela passera par la modélisation du problème à résoudre ainsi que des solutions choisies. Ainsi, nous délimiterons les parties ń boites noires ż du logiciel client et de la couche inférieure du driver et nous modéliserons la couche faisant le lien entre ces dernières. Enfin, nous élaborerons le jeu d'essai auquel

sera soumis le produit final.
Le dossier présenté à la fin de cette étape comprendra :
- un ou plusieurs modèles détaillés (de préférence UML)
- la description de l'architecture logicielle retenue
- une description précise des classes et des méthodes à implémenter
- un nouveau planning révisé du travail restant.

Etape nř3 : Réalisation

Du : 16 février 2008
Au : 23 mai 2008

Au cours de cette dernière étape, nous réaliserons les modifications nécessaires sur le driver telles que prévues lors de la partie conception. Suite à cela, nous vérifierons si le driver final répond bien aux jeux de test définis précédemment.
Seront remis au tuteur entreprise les documents suivants :
- un dossier de tests
- un manuel d'utilisation
- un manuel d'installation
- le planning révisé en fin de projet
- un bilan récapitulatif du projet.

Tout au long du projet

Chaque semaine, une fiche récapitulative du travail effectué et prévu sera remise aux tuteurs du projet. Ces fiches seront jointes aux rapports rédigés à la fin de chaque étape.
Les sources du projets seront placées sur un serveur de type CVS qui sera accessible au binôme en charge du projet ainsi qu'à leur tuteurs.
Toute documentation produite sera générée en format PDF en utilisant Latex.

Bibliographie

Les premiers drivers proposés pour commencer l'étude étaient *ODBC*[1][2] et *OLE DB*[7][8].
Les sites, qui concernaient ces deux drivers, nous ont incité à en chercher de nouveaux, car ceux-ci étaient mal adaptés à l'utilisation de Java.

Ils nous ont ensuite mené à suivre la piste de *JDBC*[3][4][5] et *ADO*[9].
Les utilisations d'*ADO* et d'*OLE DB* ne nous ont pas paru adaptées. Bien que ceux-ci soient plus performants, ils sont plus récents et donc moins utilisés.
De part sa prédestination au langage Java, le protocole *JDBC* nous a paru plus intéressant. Celui-ci semble être utilisable sur tous les logiciels possédant une interface *ODBC*.
Aussi, Java étant un langage portable, il possède des adhérents dans le monde du libre. Nous espérons donc plus facilement trouver le driver cherché en choisissant ce protocole.

A cette bibliographie, est venu s'ajouter un ouvrage de Reese, George, Soulard, Hervé[6] portant sur *JDBC*.

Quant au travail de bibliographie à effectuer, il devra orienter cette bibliographie en fonction des problèmes à résoudre. Maintenant que le type de driver a été choisi, la bibliographie devra se tourner vers le driver JDBC.

Bibliographie

[1] http://www.commentcamarche.net/odbc/odbcintro.php3

[2] http://fr.wikipedia.org/wiki/Open_database_connectivity

[3] http://www.commentcamarche.net/jdbc/jdbcintro.php3

[4] http://java.sun.com/javase/technologies/database/

[5] http://developers.sun.com/javadb/

[6] *JDBC et Java : guide du programmeur* Par Reese, George, Soulard, Hervé

[7] http://fr.wikipedia.org/wiki/OLE_DB

[8] http://msdn2.microsoft.com/fr-fr/library/0c7e23ba-03a2-41b5-8695-5e1694ec5b95.aspx

[9] http://fr.wikipedia.org/wiki/ActiveX_Data_Objects

Utilisation d'un driver ODBC sous *Access 97*

Dans l'un de ses *mails*, M. Guérin nous a communiqué une procédure à suivre, appuyée de copies d'écran, pour importer des données provenant d'une source externe vers *Access 97* à l'aide d'un driver ODBC, c'est-à-dire pour lier une table externe à une base *Access* via ODBC (c'est le même principe sous *Excel* ou un autre logiciel sous *Windows*).

1. D'abord, on ouvre le module de connexion à un structure de données.
2. On indique à Access que l'on souhaite lire une structure de données accessible via ODBC.
3. On veut sélectionner la source de données ODBC. Le moyen proposé sur cette copie d'écran est de passer par un fichier.
4. On sélectionne l'onglet « Source de données machine » afin de sélectionner le driver que nous avons créé.

   Une fois sélectionné, notre driver devrait demander sur quelle fichier et quelle table il doit se connecter, un peu comme le fait le driver « *MS Access 97 Database* ».

Voici la procédure expliquée en images :

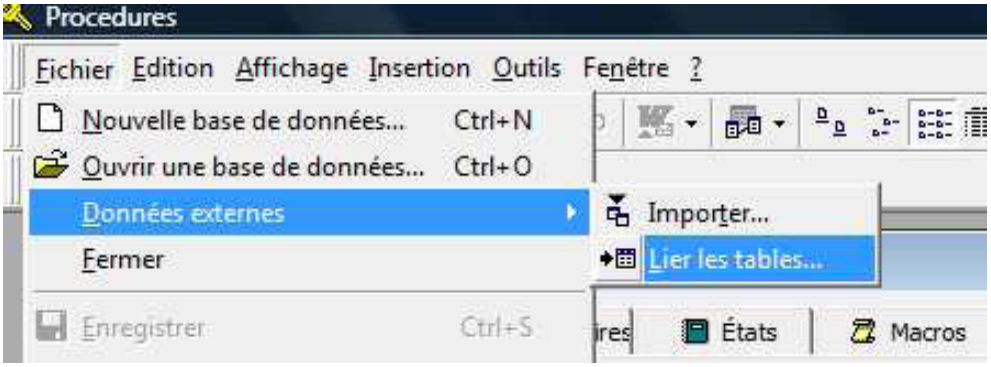


FIG. 10.1 – Etape 1 du chargement d'un driver ODBC sous *Access 97*

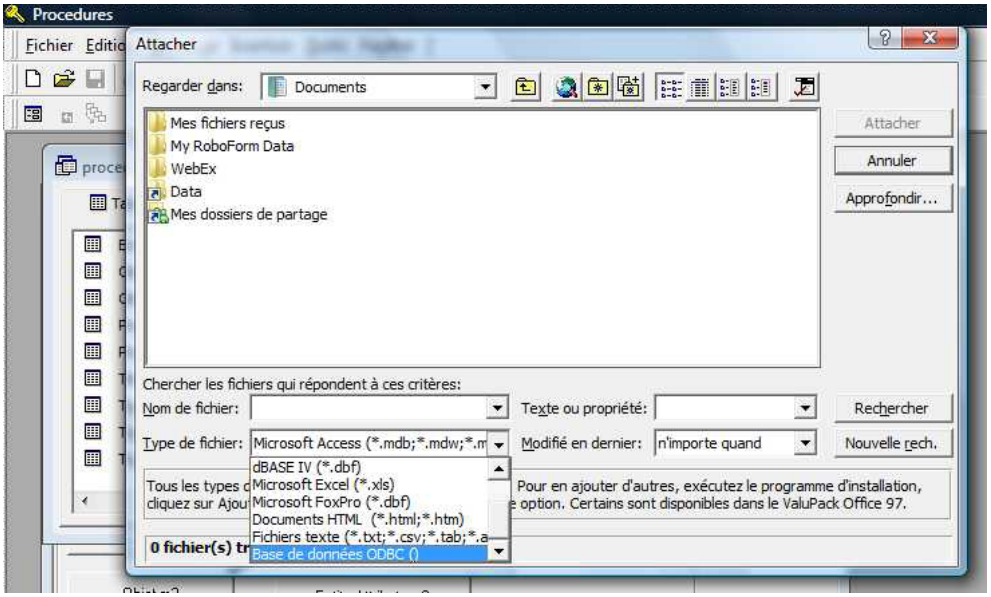


FIG. 10.2 – Etape 2 du chargement d'un driver ODBC sous *Access 97*

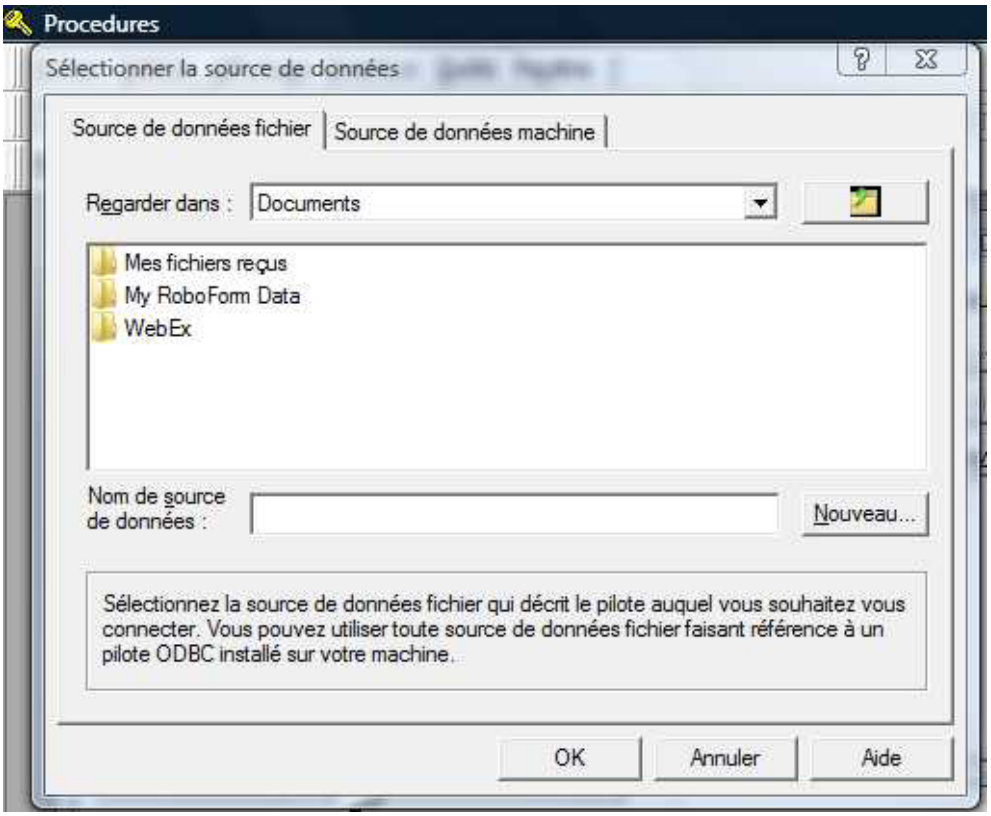

Procedures
Sélectionner la source de données
Source de données fichier
Source de données machine
Regarder dans :
Documents
Mes fichiers reçus
My RoboForm Data
WebEx
Nom de source de données :
Nouveau...
Sélectionnez la source de données fichier qui décrit le pilote auquel vous souhaitez vous connecter. Vous pouvez utiliser toute source de données fichier faisant référence à un pilote ODBC installé sur votre machine.
OK
Annuler
Aide

FIG. 10.3 – Etape 3 du chargement d'un driver ODBC sous *Access 97*

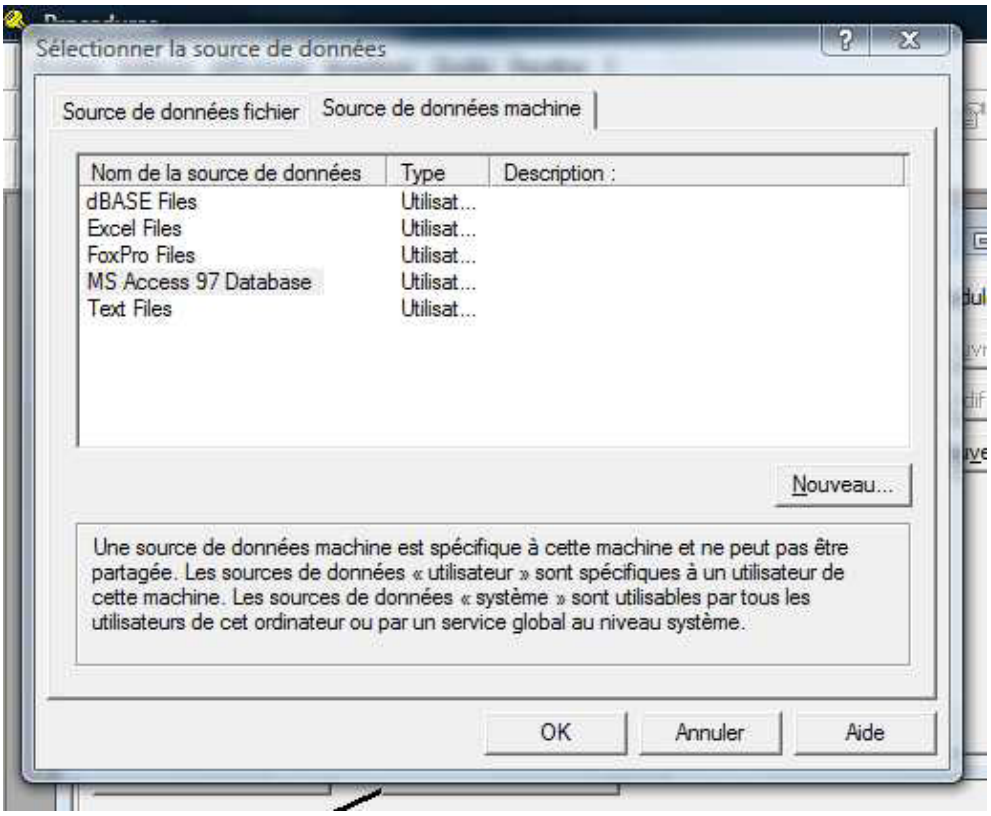

Sélectionner la source de données
Source de données fichier
Source de données machine
Nom de la source de données
Type
Description :
dBASE Files
Utilisat...
Excel Files
Utilisat...
FoxPro Files
Utilisat...
MS Access 97 Database
Utilisat...
Text Files
Utilisat...
Nouveau...
Une source de données machine est spécifique à cette machine et ne peut pas être partagée. Les sources de données « utilisateur » sont spécifiques à un utilisateur de cette machine. Les sources de données « système » sont utilisables par tous les utilisateurs de cet ordinateur ou par un service global au niveau système.
OK
Annuler
Aide

FIG. 10.4 – Etape 4 du chargement d'un driver ODBC sous *Access 97*

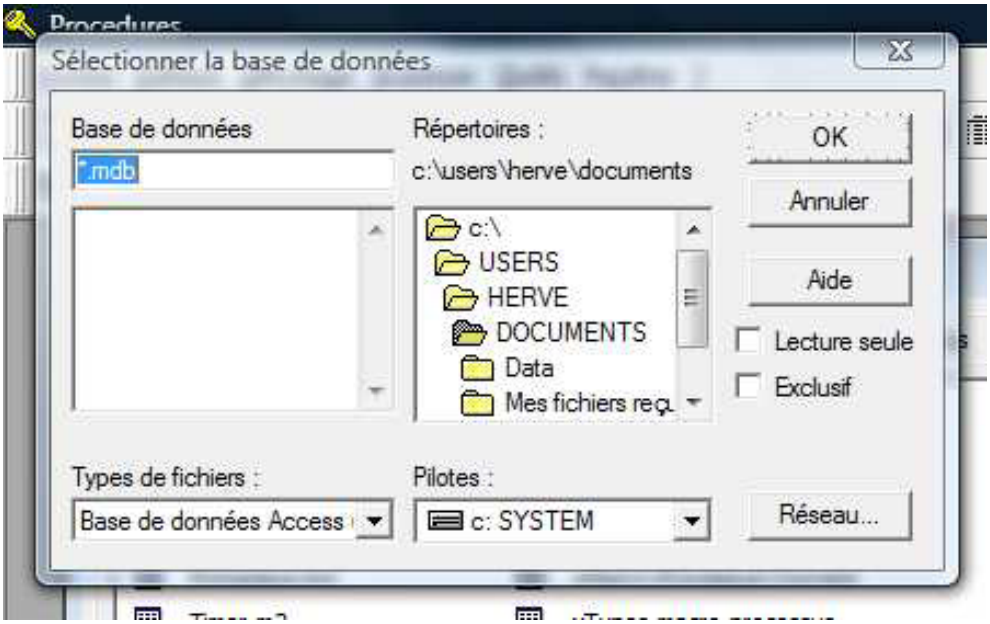


FIG. 10.5 – Etape 5 du chargement d'un driver ODBC sous *Access 97*

Rapports hebdomadaires

Fiche de suivi : Semaine du 01 au 05 Octobre 2007

Travail effectué

- Prise de conact avec le tuteur enseignant le 02 octobre pour préciser les modalités de contact du tuteur entreprise, M. Guérin.
- Prise de RDV par mail. Entretien fixé au jeudi 11 octobre à 15h.

Fiche de suivi : Semaine du 08 au 12 Octobre 2007

Travail effectué

- Entretien effectué le 11 octobre, en présence de MM. Guérin et Normand
- Etude du fonctionnement de Latex

Compte-rendu de la réunion du jeudi 11 octobre 2007

Brève présentation sur le devenir de l'outil demandé

La finalité de ce projet est de réaliser une communication entre un SGBD propriétaire, créé par notre tuteur entreprise, et un outil de requête du marché (exemple : Access, Excel, ...). Notre travail est donc de trouver un driver existant, traitant des fichiers plats ou un format propriétaire connu, de le comprendre et de le modifier, afin que celui-ci puisse être interfaçable avec la structure de données propriétaire donnée par notre tuteur entreprise.

Cet éclairage sur le début du projet nous a fourni des clés pour commencer la partie bibliographique.

Étapes

- Etude des protocoles de communication entre SGBD et outil de requêtes (ODBC ? OLE ? Autres ?)
- Recherche d'un driver simple, existant dans la communauté libre, permettant de lire dans des fichiers plats ou dans des structures de données.
- Etude de la licence GPL et de ses contraintes dans un projet à but commercial
- Etude approfondie du driver et du protocole choisis
- Adaptation du driver à la structure de données propriétaire

Modalités sur les comptes-rendus hebdomadaires et sur la manière de tenir le projet (données par M. Normand)

- Fiches hebdomadaires avec le travail effectué au cours de la semaine, le travail prévu pour la semaine à venir ainsi que les questions à poser aux tuteurs
- Travail à déposer sur un serveur CVS

Travail à effectuer

- Recherche de documentation sur ODBC
- Recherche d'un driver ODBC libre de droit

Fiche de suivi : Semaine du 15 au 19 Octobre 2007

Travail effectué

- Recherche sur les différents drivers disponibles pour travailler avec des SGBD.
- Protocoles étudiés : *ODBC*, *JDBC*, *OLE DB* et *ADO*
- Notre choix se porterait sur JDBC (voir le récapitulatif ci-joint).

Question :

- Approuvez-vous notre choix de *JDBC* comme driver pour poursuivre ce projet ?

Travail à effectuer

- Rédaction du rapport de début de projet
- Approfondir les recherches sur le protocole choisi

Fiche de suivi : Semaine du 22 au 26 Octobre 2007

Travail effectué

- Réponse à la question posée par le tuteur entreprise au sujet de JDBC
- Rédaction du compte-rendu de l'entretien du jeudi 11 octobre
- Rédaction du rapport de début de projet. Celui-ci contient le cahier des charges, ainsi qu'une ébauche de bibliographie.

Travail à effectuer

- Continuer les recherches sur le protocole choisi
- Se renseigner sur la communauté du libre travaillant sur JDBC

Fiche de suivi : Semaine du 05 au 09 novembre 2007

Travail effectué

Nous avons bien pris en compte les remarques formulées par notre tuteur entreprise sur le cahier des charges.
Pour répondre à l'une de ses questions : JDBC est capable de gérer plus de 100 SGBD différents dont les plus utilisés : MS Access, Oracle, MySQL, ...

Il existe à ce jour 4 types de drivers JDBC, ne gérant pas tous les mêmes SGBD :

1. Le bridge JDBC-ODBC : il permet l'accès au SGBD via le driver ODBC.
2. Native-API partly-Java driver : ce type de driver traduit les appels de JDBC à un SGBD particulier, grâce à un mélange d'API java et d'API natives.
3. JDBC-Net pure Java driver : ce type de driver (écrit entièrement en java) passe par un serveur intermédiaire pour l'accès au SGBD.
4. Native-protocol pure Java driver : ce type de driver (écrit entièrement en java) se connecte directement au SGBD.

A première vue, notre choix se porterait sur le premier driver (pont JDBC-ODBC), qui permet la communication avec tous les SGBD et logiciels de traitement de données ayant une interface ODBC (tels que Excel, Access, Oracle, ...).

Bien que nous ne connaissions pas l'étendue des possibilités des autres pilotes, nous ne les écarterons pas de nos recherches. Ceux-ci étant prisés par les adeptes du libre (70% des pilotes SGBD sont du types 3 et 4), il nous sera plus facile de trouver un driver exploitable de ces types.

Travail à effectuer

- Continuer les recherches sur le protocole choisi
- Se renseigner sur la communauté du libre travaillant sur JDBC

Fiche de suivi : Semaine du 12 au 16 novembre 2007

Travail effectué

- Prise de contact avec notre tuteur entreprise
- Début de recherche sur les drivers, ce qui nous a permis de connaître les difficultés que nous pourrons rencontrer

Compte-rendu de l'entretien du 13/11/2007

- Nous nous sommes mis d'accord sur les types de drivers que nous pourrions utiliser pour la suite du projet.
- Nous avons vu plus en détail les besoins du tuteur entreprise. Au cours de cette réunion, il a été convenu que des documentations sur les licences *Open Source* et sur les drivers devraient être réalisées et présentées dans la bibliographie.
- Pour finir cette réunion, nous avons pris conseil auprès de notre tuteur entreprise pour savoir comment aborder la recherche d'un driver *Open Source*.

Travail à effectuer

- Objectif : faire une documentation sur les licences *Open Source*
- Commencer une documentation sur les drivers : mode de fonctionnement, structure, ...
- Se renseigner sur la communauté du libre travaillant sur JDBC

Fiche de suivi : Semaine du 19 au 23 novembre 2007

Travail effectué

- Prise de contact avec notre tuteur enseignant
- Réalisation d'une documentation sur les licences libres
- Début de la réalisation d'une documentation sur les drivers : mode de fonctionnement, structure, ...

Contact avec M. Normand, en présence de M. Ricordel

- Demande de renseignements sur le travail à rendre dans le cadre de l'étude "*Dimension commerciale et/ou organisationnelle du projet*"
- Demande de renseignements sur les communautés du libre existantes, ainsi que sur les licences
- M. Ricordel, présent lors de cette entretien, nous a conseillé de nous renseigner sur la licence BSD, qui semble être la mieux adaptée à notre projet

Travail à effectuer

- Prendre contact avec le tuteur entreprise, afin d'obtenir de plus amples informations sur son projet, dans le but de réaliser une étude sur la "*Dimension commerciale et/ou organisationnelle du projet*"
- Se renseigner sur la communauté travaillant sur les licences libres (FreshMeat, SourceForge, ...)
- Se renseigner sur la communauté du libre travaillant sur JDBC

Fiche de suivi : Semaine du 26 au 30 novembre 2007

Travail effectué

- Suite de la réalisation d'une documentation sur les drivers : mode de fonctionnement, structure, ...
- Approche des sites sur les licences libres (FreshMeat, SourceForge, ...)
- Réunion avec Jacques Moraud et Dominique Pécaud, en rapport en rapport avec l'étude "*Dimension commerciale et/ou organisationnelle du projet*"
- Prise de contact avec le tuteur entreprise, afin d'obtenir de plus amples informations sur son projet, dans le but de réaliser cette étude

Réunion avec Jacques Moraud et Dominique Pécaud

- Demande de renseignements sur le travail à rendre dans le cadre de l'étude "*Dimension commerciale et/ou organisationnelle du projet*"
- Avant le 14 décembre, doivent être remis à Jacques Moreau une présentation rapide du projet et de l'entreprise (une page maximum) ainsi qu'une thématique que nous souhaitons développer

Travail à effectuer

- Avant le 14 décembre : faire valider par Jacques Moreau l'étude Marketing
- La soutenance concernant la bibliographie du projet transversal se déroulera le mardi 18 décembre, à 10h00, en salle C008, en présence de MM. Jean-Pierre Guédon et Nicolas Normand
- Se renseigner sur la communauté travaillant sur les licences libres (FreshMeat, SourceForge, ...)
- Se renseigner sur la communauté du libre travaillant sur JDBC

Fiche de suivi : Semaine du 03 au 07 décembre 2007

Travail effectué

- Fin de la réalisation d'une documentation sur les drivers : mode de fonctionnement, structure, ...
- Approche des sites sur les licences libres (FreshMeat, SourceForge, ...)
- Réunion avec le tuteur entreprise, afin d'obtenir de plus amples informations sur son projet, dans le but de réaliser cette étude

Réunion avec Mr. Guérin

- Demande de renseignements sur le travail à rendre dans le cadre de l'étude "*Dimension commerciale et/ou organisationnelle du projet*"
- Demande de renseignements sur la structure des composants entrant en jeu dans le projet

Travail à effectuer

- Avant le 14 décembre : faire valider par Jacques Moreau l'étude Marketing
- La soutenance concernant la bibliographie du projet transversal se déroulera le mardi 18 décembre, à 10h00, en salle C008, en présence de MM. Jean-Pierre Guédon et Nicolas Normand
- Se renseigner sur la communauté du libre travaillant sur JDBC
- Changer le format LaTeXdes fiches hebdomadaires

Fiche de suivi : Semaine du 10 au 14 décembre 2007

Travail effectué

- Réalisation de l'ébauche de l'étude commerciale du projet
- Réalisation du support informatique pour la soutenance orale
- Préparation à la soutenance orale

Travail à effectuer

- La soutenance concernant la bibliographie du projet transversal se déroulera le mardi 18 décembre, à 10h00, en salle C008, en présence de MM. Jean-Pierre Guédon et Nicolas Normand
- Effectuer un test sur Excel d'un driver JDBC

Fiche de suivi : Semaine du 15 au 21 décembre 2007

Travail effectué

- Soutenance orale de l'étape "Bibliographie et analyse des besoins" du projet transversal, en présence de MM. Hervé Guérin, Nicolas Normand et Jean-Pierre Guédon
- Entretien avec M. Hervé Guérin afin de préciser les objectifs qui débuteront l'étape "Conception"

Travail à effectuer

- Analyser un driver JDBC
- Effectuer un test (sur Excel, ...) d'un driver JDBC

Fiche de suivi : Semaine du 07 au 11 janvier 2008

Travail effectué

- Installation d'un driver ODBC (MyODBC pour MySQL) et test sous Excel pour se familiariser avec l'interface ODBC
- Recherche d'informations sur les drivers JDBC-ODBC
- Demande d'aide spécifique sur les forums de deux sites d'entraide de développeurs : CommentCaMarche.net et Developpez.net

Travail à effectuer

- Déterminer précisément une solution pour réaliser l'exportation de données vers les SGBD
- Commencer l'analyse d'un driver open-source

Fiche de suivi : Semaine du 14 au 18 janvier 2008

Travail effectué

- Poursuite de la recherche d'informations concernant les drivers JDBC-ODBC, sur les sites suivants :
  - http://www.developpez.net
  - http://www.cppfrance.com
  - http://msdn2.microsoft.com/en-us/default.aspx
- Mise en ligne de demandes sur les forums de sites d'entraide de développeurs : pour l'instant, peu d'informations nous parviennent

Travail à effectuer

- Approfondir la recherche d'informations sur les drivers JDBC et ODBC
- Chercher à déterminer une solution afin de réaliser l'exportation de données vers les SGBD
- Commencer l'analyse d'un driver open-source

Fiche de suivi : Semaine du 21 au 25 janvier 2008

Travail effectué

- Lors de nos recherches d'informations sur les drivers JDBC et ODBC, nous avons pris connaissance d'un terme pouvant décrire la communication d'un driver ODBC vers JDBC : *ODBC to JDBC Gateway*
- Nous avons trouvé plusieurs drivers qui pourraient nous aider à mieux comprendre une telle architecture :
  - depuis FreshMeat : EasySoft, OpenLink Software et Basis qui sont dotés d'une licence propriétaire avec démonstration de quelques jours gratuite
  - depuis SourceForge : le seul projet ODBC-JDBC est un « projet mort »
- Une deuxième solution que nous envisageons serait de créer un driver ODBC compatible avec Java via *JNI*

Travail à effectuer

- Continuer la recherche de solutions (*ODBC to JDBC Gateway* et *ODBC/JNI*)

Fiche de suivi : Semaine du 28 janvier au 1er février 2008

Travail effectué

- Nous avons continué nos recherches sur les drivers ODBC-JDBC. Cette technologie semblerait bien être une solution viable pour le projet. Cependant, nos recherches sur cette solution ont été peu fructueuses.
- Nous avons également poussé nos recherches sur la solution ODBC-JNI. Nous n'avons cependant trouvé aucune information à ce sujet.
- Nous avons recherché, téléchargé puis étudié différents drivers (ODBC-JDBC et ODBC). Voici quelques sources de driver ODBC correspondant à notre attente, mais peu documentés et peu commentés :
  - Open Database Transport Protocol
  - iODBC
- ODBC-JDBC Project est un projet "mort" de SourceForge qui, malgré l'absence de code source, rassemble une documentation intéressante au sujet d'ODBC, JDBC et JNI.
- Le driver ODBC-JDBC du projet "Firebird" est bien documenté mais ses sources sont peu commentées. Nous ne sommes pas surs de la validité de la licence du code source pour un projet commercial.

Travail à effectuer

- Continuer l'étude du driver ODBC-JDBC du projet "Firebird"
- La soutenance concernant la modélisation du projet transversal se déroulera le vendredi 22 février 2008 à 08h30 en salle C008, en présence de MM. Jean-Pierre Guédon et Nicolas Normand

Fiche de suivi : Semaine du 04 au 08 février 2008

Travail effectué

- Réunion, mercredi 06 février, en présence de MM. Hervé Guérin et Nicolas Normand
- Début d'étude du site MSDN

Compte-rendu de la réunion

- Précision des contraintes du projet
- Etude des fonctions disponibles dans la documentation de MSDN
- Utilisation de SDK afin de créer notre driver ODBC
- Etude du driver de lecture de fichiers plats (`txt`, `csv`, ...)

Travail à effectuer

- Poursuivre l'étude du SDK de Microsoft et du site MSDN
- La soutenance concernant la modélisation du projet transversal se déroulera le vendredi 22 février 2008 à 08h30 en salle C008, en présence de MM. Jean-Pierre Guédon et Nicolas Normand

Fiche de suivi : Semaine du 11 au 15 février 2008

Travail effectué

- Réalisation du rapport de conception du projet
- Poursuite des diverses études engagées

Travail à effectuer

- Réaliser le support informatique nécessaire à la soutenance orale
- Se préparer à la soutenance orale

Fiche de suivi : Semaine du 18 au 22 février 2008

Travail effectué

- Analyse du driver « GODBC »
- Réalisation du support informatique pour la soutenance orale
- Réalisation de la soutenance orale le vendredi 22 février 2008 à 09h00, dans la salle C008, en présence de M. Jean-Pierre Guédon, Hervé Guérin et Nicolas Normand
- Entretien avec M. Hervé Guérin le midi suivant la soutenance

Compte-rendu de la réunion

Entretien avec Hervé Guérin, le vendredi 22 février 2008 à 12h50, au cours duquel nous nous sommes mis d'accord sur les points suivants :

- les requêtes prises en charge par l'interpréteur SQL
- les caractéristiques du *thread* serveur et ses interactions avec le logiciel serveur
- les étapes importantes de la partie réalisation

Travail à effectuer

- Installation et test de l'exemple de driver « GODBC »

Fiche de suivi : Semaine du 03 au 07 mars 2008

Travail effectué

- Tentatives d'installation du driver squelette ODBC. Problème : nous n'arrivons toujours pas à créer une source de données depuis ce driver
- Analyse des différentes fonctions de l'API ODBC

Travail à effectuer

- Envoyer un message au concepteur du driver squelette
- Créer l'exécutable d'installation

Fiche de suivi : Semaine du 10 au 14 mars 2008

Travail effectué

- Envoi d'un message au rédacteur de l'article sur « GOBDC »
- Emprunt et début de lecture du livre « Inside ODBC »
- Analyse des différentes fonctions de l'API ODBC

Travail à effectuer

- Créer l'exécutable d'installation

Fiche de suivi : Semaine du 17 au 21 mars 2008

Travail effectué

- Pas de réception du message envoyé au rédacteur de l'article sur « GOBDC »
- Poursuite de la lecture du livre « Inside ODBC »
- Recherche d'un nouveau driver ODBC simple sur Internet

Travail à effectuer

- Créer l'exécutable d'installation
- Envoyer un second mail de demande d'aide au rédacteur de l'article sur « GODBC »
- Revoir notre planification des tâches en fonction des difficultés rencontrées

Fiche de suivi : Semaine du 24 au 28 mars 2008

Remarque

Suite aux difficultés rencontrées (impossibilité de faire fonctionner le driver squelette), nous prenons un lourd retard dans la réalisation du driver.

Nous explorons actuellement d'autres pistes (livres "Inside ODBC") afin de contourner le problème. Le livre que nous sommes en train d'étudier étant en anglais, son étude sera longue et nous ne pourrons avancer aussi rapidement que prévu.

Travail effectué

- Envoi d'un second mail de demande d'aide au rédacteur de l'article sur « GOBDC »
- Poursuite de la lecture du livre « Inside ODBC »
- Recherche d'un nouveau driver ODBC simple sur Internet sans succès

Travail à effectuer

- Créer l'exécutable d'installation
- Revoir notre planification des tâches en fonction des difficultés rencontrées

Fiche de suivi : Semaine du 31 mars au 04 avril 2008

Travail effectué

- Pas de réception du second message envoyé au rédacteur de l'article sur « GOBDC »
- Recherche d'un nouveau driver ODBC simple sur Internet

Travail à effectuer

- Réunion prévue le lundi 07 avril 2008, à partir de 14h00

Fiche de suivi : Semaine du 07 au 11 avril 2008

Travail effectué

- Réunion effectuée le lundi 07 avril 2008, à partir de 14h00
- Diffiérents tests et recherches sur le driver ODBC afin de trouver de la source de l'erreur : il semblerait qu'une option non-activée du logiciel empêche la bonne compilation

Résumé de la réunion

- On note des différences entre le fonctionnement du driver compilé par nous-mêmes depuis les sources, et le driver fourni au format DLL
- Contrairement à la déduction effectuée pendant la réunion, il n'y a pas de différences d'exécution du driver entre *Windows XP* et *Windows Vista*

Travail à effectuer

- Tests sur d'autres compilateurs (*Visual Studio*, *Eclipse*, ...)
- Réaliser un suivi étape par étape des fonctions appelées par le pilote
- Utiliser le suivi pour demander de l'aide sur des forums de développeurs

Fiche de suivi : Semaine du 28 avril au 02 mai 2008

Travail effectué

- Rédaction de l'étude commerciale du projet

Travail à effectuer

- La soutenance du projet aura lieu le lundi 26 mai 2008 à 16h00, en salle B202

Fiche de suivi : Semaine du 05 au 09 mai 2008

Travail effectué

- Nouveaux tests réalisés pour le driver squelette ODBC
- Envoi du compte-rendu de l'étude commerciale du projet à M. Jacques Moreau
- Début de rédaction du rapport de réalisation
- Début de rédaction du support vidéo pour la soutenance orale

Travail à effectuer

- Continuer la rédaction du rapport de réalisation
- Continuer la rédaction du support vidéo pour la soutenance orale
- La soutenance du projet aura lieu le lundi 26 mai 2008 à 16h00, en salle B202

Fiche de suivi : Semaine du 12 au 16 mai 2008

Travail effectué

- Nouveaux tests de compilation sur le driver squelette ODBC
- Poursuite de la rédaction du rapport de réalisation
- Poursuite de la rédaction du support vidéo pour la soutenance orale

Travail à effectuer

- Poursuivre la réalisation du pilote
- Terminer la rédaction du rapport de réalisation
- Achever la rédaction du support vidéo pour la soutenance orale
- La soutenance du projet aura lieu le lundi 26 mai 2008 à 16h00, en salle B202

Fiche de suivi : Semaine du 19 au 23 mai 2008

Travail effectué

- Nouveaux tests de compilation sur le driver squelette ODBC
- Fin de la rédaction, et envoi du rapport de réalisation
- Poursuite de la rédaction du support vidéo pour la soutenance orale

Travail à effectuer

- Poursuivre la réalisation du pilote
- Terminer la rédaction du support vidéo pour la soutenance orale
- La soutenance du projet aura lieu le lundi 26 mai 2008 à 16h00, en salle B202

Bibliographie

[1] CommentCaMarche.net. Introduction sur les drivers
. *http://www.commentcamarche.net/drivers/drivers.php3*.

[2] CommentCaMarche.net. Comemnt installer jdbc sous openoffice
. *http://www.commentcamarche.net/forum/affich-2212179-installer-le-pilote-jdbc*.

[3] CommentCaMarche.net. Introduction sur les drivers jdbc
. *http://www.commentcamarche.net/jdbc/jdbcintro.php3*.

[4] Informations techniques sur jdbc
. *http://www.dil.univ-mrs.fr/~massat/ens/java/jdbc.html*.

[5] George Reese Hervé Soulard. Jdbc et java : guide du programmeur (2ème édition)
. 2001.

[6] Wikipedia. Article sur jdbc (type 1)
. *http://en.wikipedia.org/wiki/JDBC_type_1_driver*.

[7] FreshMeat. Drivers jdbc disponibles
. *http://freshmeat.net/search/?q=jdbc\&section=projects*.

[8] Site web de SUN. Drivers jdbc
. *http://developers.sun.com/product/jdbc/drivers/*.

[9] Wikipedia. Article sur jdbc
. *http://en.wikipedia.org/wiki/JDBC_driver*.

[10] Wikipedia. Article sur le logiciel libre
. *http://fr.wikipedia.org/wiki/Logiciel_libre*.

[11] Site web de l'AFUL. Définition du logiciel libre
. *http://www.aful.org/ressources/presentation/libre*.

[12] Site personnel. Les différentes licences open source
. *http://codesquale.googlepages.com/licencesopensource.pdf*.

[13] Site personnel. Lies différentes licences open source
. *http://louis.cova.neuf.fr/blocs-notes/page8.html*.

[14] Wikipedia. Article sur la licence publique générale gnu
. http://fr.wikipedia.org/wiki/Licence_publique_générale_GNU.

[15] Site web de GNU. Vendre des logiciels libres
. http://www.gnu.org/philosophy/selling.fr.html.

[16] Site web de GNU. Définition du logiciel libre
. http://www.gnu.org/philosophy/free-sw.fr.html.

[17] Wikipedia. Article sur la licence publique générale limitée gnu
. http://fr.wikipedia.org/wiki/Licence_publique_générale_limitée_GNU.

[18] Site web d'Eclipse. Licence eclipse
. http://www.eclipse.org/org/documents/epl-v10.php.

[19] Wikipedia. Article sur cvs
. http://fr.wikipedia.org/wiki/Concurrent_versions_system.

[20] Wikipedia. Article sur subversion
. http://fr.wikipedia.org/wiki/Subversion_(logiciel).

[21] Site web d'OpenLink Software. Driver odbc-jdbc
. http://uda.openlinksw.com/odbc/.

[22] Site web de SUN. Documentation sur jni
. http://java.sun.com/docs/books/jni/html/intro.html.

[23] Site web d'Easysoft. Driver odbc-jdbc
. http://www.easysoft.com/products/data_access/odbc_jdbc_gateway/.

[24] Site web de SourceForge. Driver odbc-jdbc (« dead project »)
. http://odbcjdbc.sourceforge.net/.

[25] Site web de Firebird. Driver odbc-jdbc
. http://www.praktik.km.ua/.

[26] Site web d'OpenAccess. Driver odbc-jdbc
. http://www.openaccesssoftware.com/.

[27] Site web de iODBC. Connecteur odbc/mysql
. http://www.iodbc.org/index.php?page=mysql2odbc/index.

[28] Site web de MySQL. Driver odbc
. http://dev.mysql.com/downloads/connector/odbc/3.51.html.

[29] Site web de Microsoft. Sdk odbc
. ftp://ftp.microsoft.com/developr/ODBC/public/.

[30] Site web de Microsoft. Msdn odbc
. *http://msdn2.microsoft.com/en-us/library/ms714562.aspx*.

[31] Expert system-designer et Expert VisualC++ Agrawal Vikash. Exemple de pilote godbc
. *http://www.ddj.com/windows/184416434?pgno=1*.